UNIVERSITÄT BASEL

Computer Science / 15731-01 / 2018

Multimedia Retrieval

Chapter 4: Basic Image, Audio, and Video Retrieval

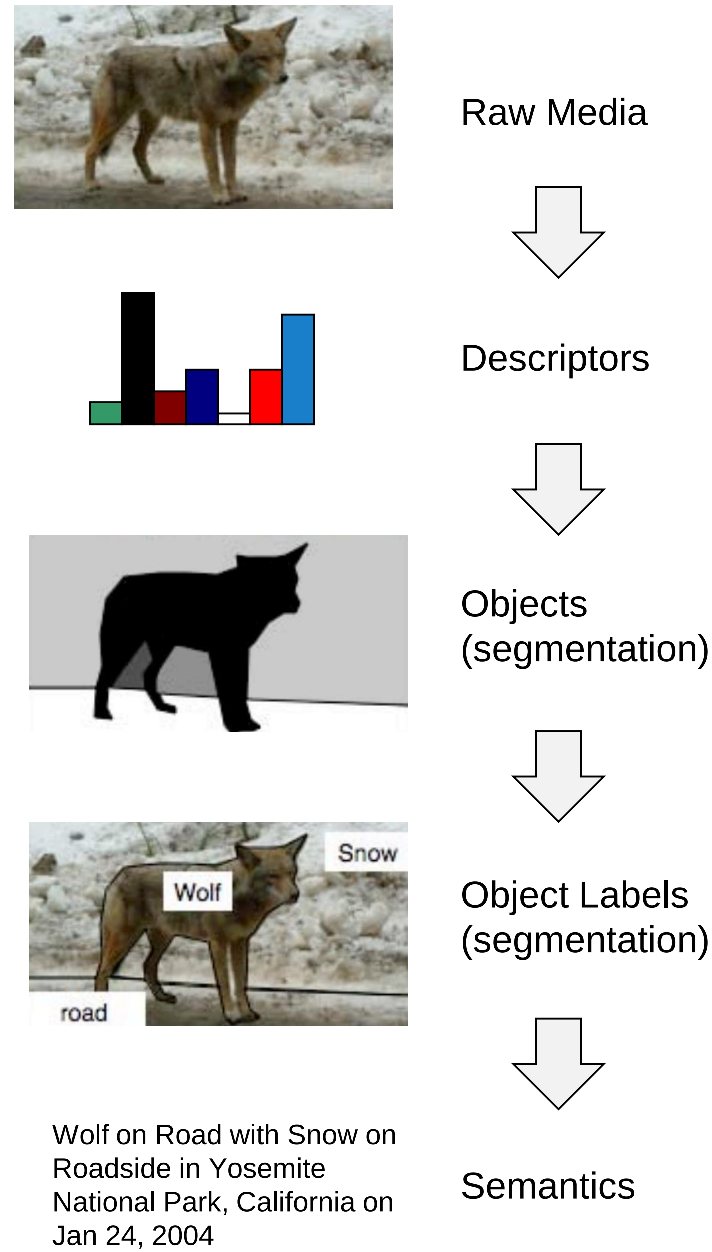
Dr. Roger Weber, roger.weber@ubs.com

4.1 Introduction

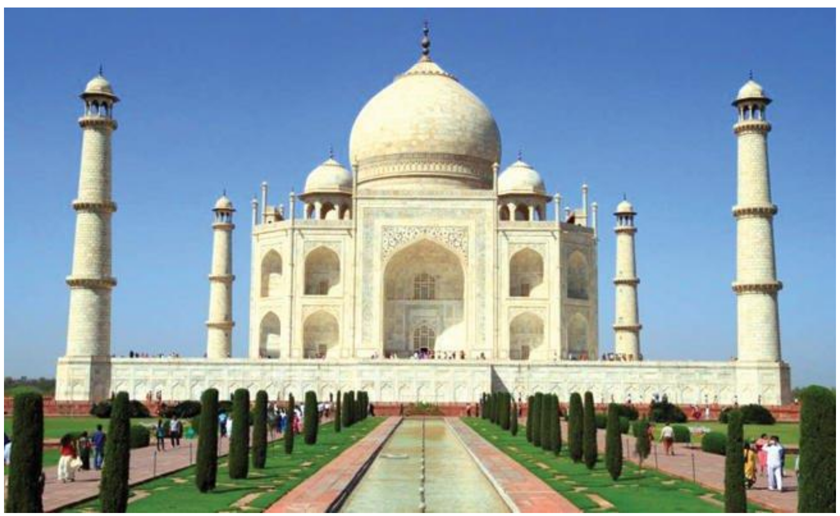
- With text and web retrieval, the descriptors for documents are the same as for user queries (words, phrases). Search performance is generally good even though we are just considering term occurrences. With other media types, it is no longer that simple. A user may want to query with natural language, but the documents do not contain keywords rather low-level signal information. This is known as the **Semantic Gap**.
  – Consider the image below. For a machine, it contains pixels each with a color code attached to it. In some cases, additional meta-information may exist. For a person, it depicts the Spalentor in Basel. When looking for the Spalentor in images, we need to translate the term "Spalentor" somehow to the low-level signal information (or vice-versa). But which patterns in the picture let a machine understand that this is a picture relevant for the query "Spalentor".
  – The semantic gap is the difference between the information extractable in an automated fashion from the raw image data and the interpretation of that same data by a person.
  – Also note that the semantic gap also depends on the person asking the question; for someone unfamiliar with Basel's history, the picture is simply an interesting piece of architecture.

- The same gap applies to audio files. A user is not expressing a query at the signal level (amplitude, frequencies, etc.) but at a semantic level: “find me a rock ballad” or “funny comedian”.
- Humans interpret signal information in several steps:
  1. Perception – we are not measuring the physical quantities but rather obtain a “biased” perception that helps us to further infer information.
     - The eye is responding to three color channels and luminance. The concept of color is merely an interpretation of our brain, but it is essential to the next steps. Both eyes combined provide a spatial perspective of the scenery.
     - The ear is responding to wave lengths and measures delays between the ears to infer direction of the sound. The pre-processed signal that reaches the brain is no longer physical quantities.
  2. Generic Semantic Inference – the brain interprets the perception and enriches it with semantic information. The first step is poorly generic and is focused on important aspects (person, animal, sky, faces). At this stage, information hiding prevents over-stimulation of reasoning.
  3. Specific Semantic Inference – with our knowledge, experience, cultural conditioning, and beliefs, we infer contextual semantics including named objects (Spalentor), events (Soccer match), and abstract concepts (emotions, spatial, time).
     - This step depends on the individual experience and knowledge of a person. You will infer different semantics for a picture of your mother than someone who does not know her.
- To close the semantic gap, a machine must address each of the three levels. Content-Based Retrieval systems started with the perceptual level. Recently, deep learning made huge progress on the generic semantics and on the specific semantics. In between, we have classical retrieval on metadata obtained either by manual or automated processes. Metadata is matching the semantics of users much better and is still the dominating search paradigm.

- A retrieval system must mimic the human's interpretation of the low-level signal
  - The raw media is mapped to low-level descriptors that summarize information on regions, color, texture, or points of interest. To be effective, we need to replicate human's perception.
  - Object recognition combines prototypical descriptors and infers regions/blobs of interest. Image segmentation yielding a number of objects but without any classification.
  - Object labeling associates classes or names to objects often using machine learning or statistical approaches. The labels correspond to the generic semantics of users but may still fail on the specific interpretation of users.
  - Semantics result from additional contextual information either derived from the objects and their relation or through meta-data and the usage of a knowledge base. The hardest part is to obtain the context (which is also not easy for humans).
- Again, the same applies to audio and video data.



- We distinguish between two feature types going forward
  - Low level features that are based on the raw signal information and describe perception rather than semantics. Most of the early Content-Based Retrieval System were focused on low-level features and search paradigms like **Query by Example**, **Query by Sketch**, or **Query by Humming**. As a general idea, these systems extract features from both the query and media objects, and perform a comparison to find best matches (similarity search, nearest neighbor search). The semantic gap is only closed with regard to perception; higher level gaps remain open and can challenge the user during the search (like this picture but need an other color for the car, or: can't sing correct but the tune is somehow like this).
  - High level features address **generic**, **specific**, and **abstract** semantic meaning. We can distinguish between **object**, **spatial**, **temporal**, and **event/activity** information. Further information encompasses **related concepts/objects**, **abstract concepts**, and **context**. For instance, let us consider the following picture of the Taj Mahal:



Object Facet	Value
Generic Object Instance	building, water, sky
Generic Object Class	mausoleum, tomb, dome, minaret
Specific Named Object Class	UNESCO World Heritage Site (since 1983)
Specific Named Object Instance	Taj Mahal

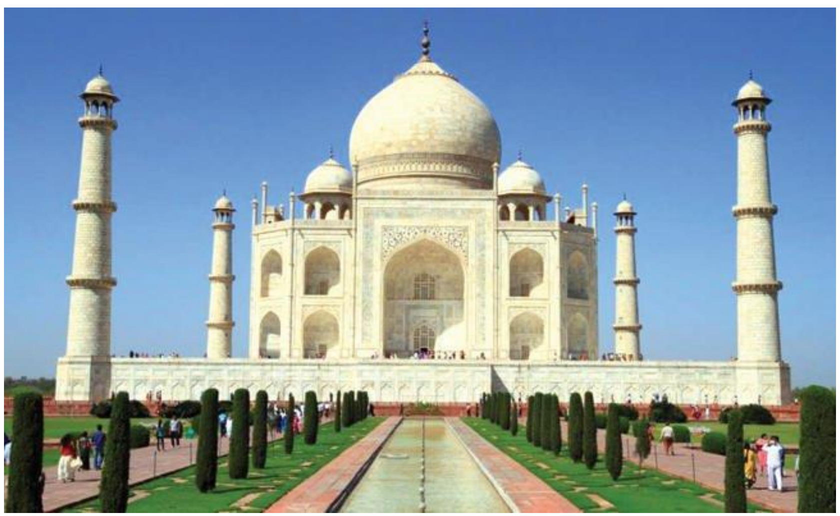
– Taj Mahal (contd)

Spatial Facet	Value
Generic Location	outside
Specific Location Hierarchy	India, Uttar Pradesh, Agra

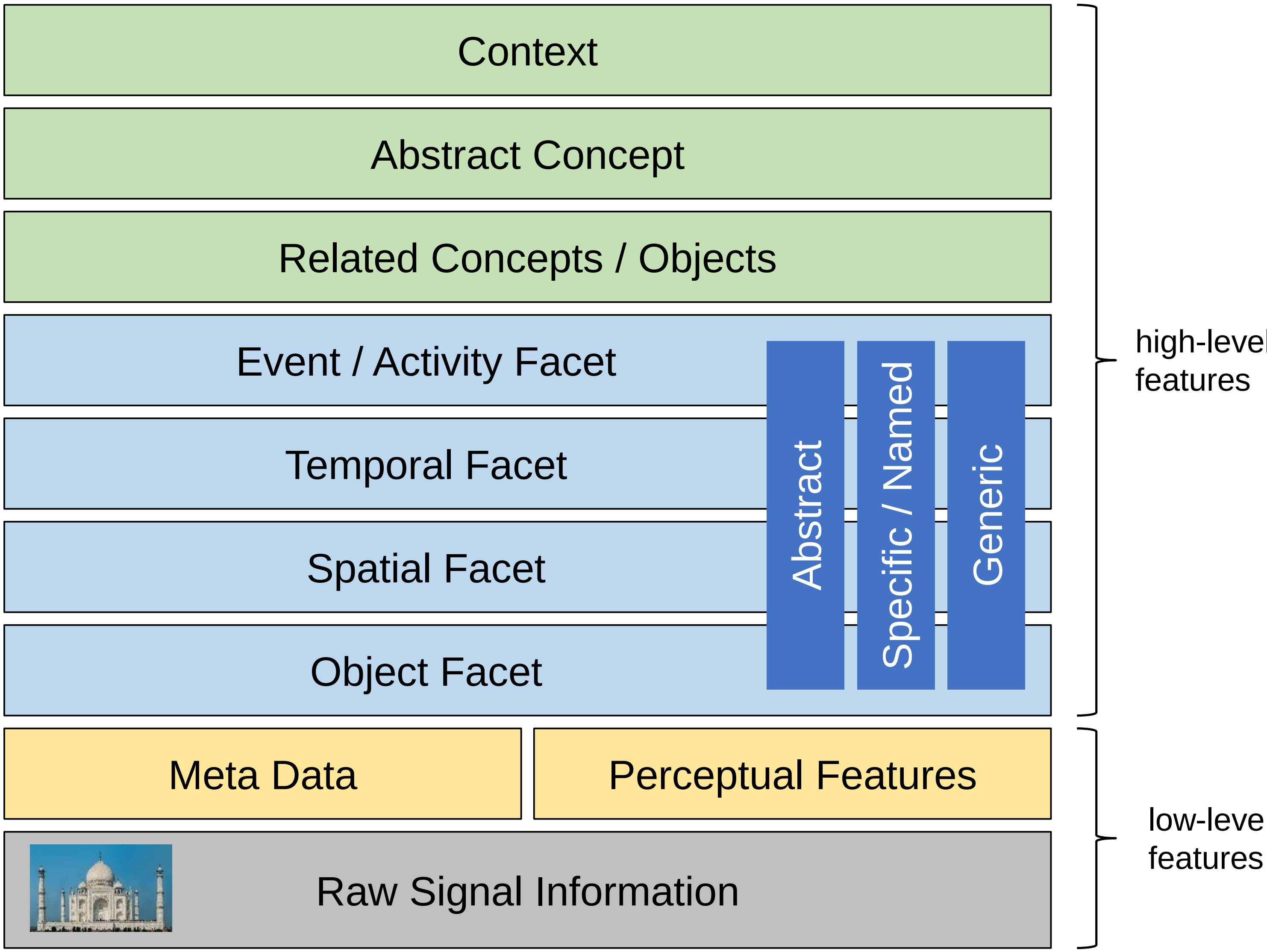
Temporal Facet	Value
Generic Time	summer, daytime
Specific Time	2006 (photo taken)

Event / Activity Facet	Value
Generic Event/Activity	tourism, attraction
Specific Event Instance	International World Heritage Expert Meeting on Visual Integrity in 2006

Contextual Facet	Value
Topic	Indian Architecture
Related Concepts / Objects	Shah Jehan, Mumtaz Mahal, Islam
Abstract Concept	love, death, devotion, remembrance
Context	built in memory of his favorite wife Mumtaz Mahal, by Shah Jehan; completed 1648

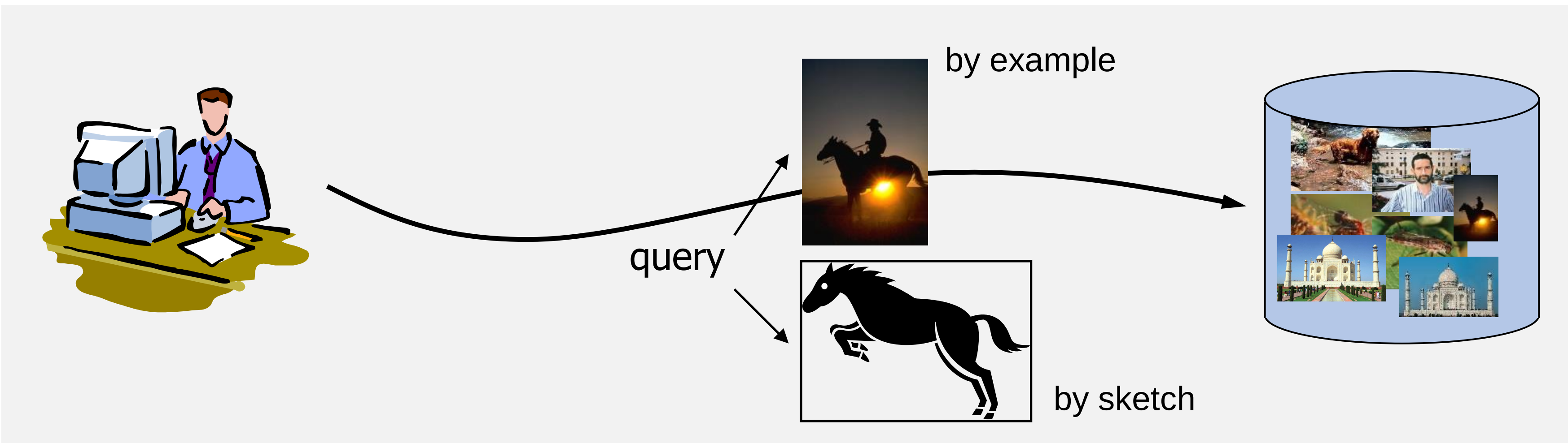


- In summary, to close the semantic gap, we need to extract descriptors at different levels allowing a user to ask semantic queries. In this chapter, we start with the lower levels. The next chapter addresses some of the higher levels.



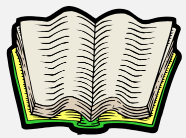
4.2 Similarity Search

- Content-based Retrieval Systems operate only with low-level features and hence struggle more with closing the semantic gap between user queries and the extracted information.
  - **Extract Meta-Data** and perform classic text or web retrieval. This is the dominant method used by most search engines on the web and multimedia repositories. The signal information is considered partially, but the focus is on key words and structural information extracted from the object or its embedding. We will consider meta-data extraction in the next section. The semantic gap is closed by automatically or manually associating key words to the media object such that the user can naturally search for objects.
  - **Query by Example / Query by Sketch (Humming)** requires the user to provide (or sketch, sing) an example of what she looks for. The example or sketch is mapped to perceptual features and search is performed based on similarity scoring in that feature space. In combination with relevance feedback, the user is able to adjust her query during the search session. The semantic gap is closed by queries in the same perceptual space.



- In the following, we briefly overview the similarity search problem (more details in Chapter 6).
  - Similarity search works on the descriptors obtained from the raw media files. We already have seen the extraction of textual features in the previous chapters. For images, audio and video files, we will study algorithms that describe a particular perceptual aspect, often in the form of a multi-dimensional feature vector. Examples:

text documents

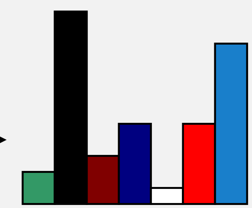


feature extraction

docID = doc10
dog → word 10, word 25
cat → word 13
home → word 2, word 27
...

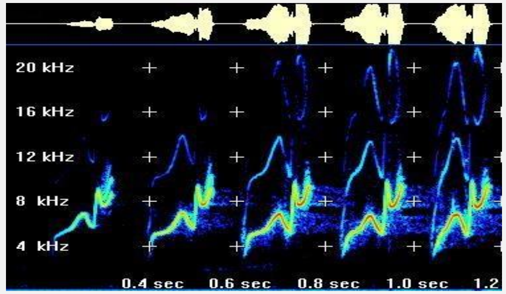
images

feature extraction



color histogram

audio files

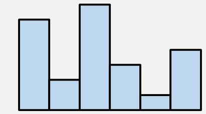


feature extraction

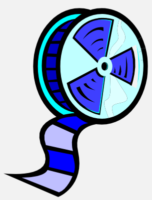
phonemes: imnOrd@namfo:rmita:gs...

text: Im Norden am Vormittag...

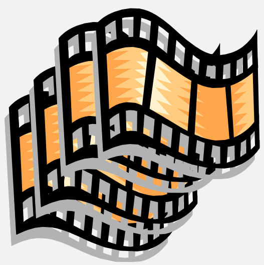
acoustical features:



video files



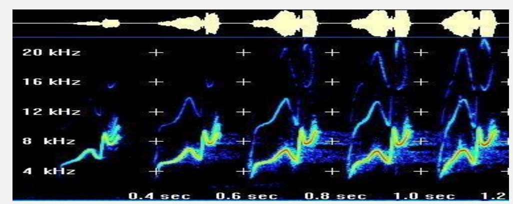
video



sequences

feature extraction

key frames



Audio Signal

subtitle: [President] I never had ....

- The definition of a similarity scoring function depends on the feature design. Hence, there is not a single measure or best-practice but individual metrics depending on the following aspects
  - **Segmentation:** we can divide a media file into segments. For instance, objects in an image, time windows in an audio or video file, sequence and shots in a video. Feature extraction either describes the entire media file (global descriptor) or apply only to segments (local/temporal descriptor). The similarity functions for local descriptors may include partial match query, while the function on global descriptors can not do so.
  - **Invariances:** feature design focuses on the extraction of robust descriptors. Robustness denotes the ability of a descriptor to remain the same (or change only little) given transformations of the original media file. For example, an image descriptor is scale invariant, if the value does not change significantly if the image is scaled up or down. Similarly, an audio descriptor is invariant to background noise, if the extracted information (e.g., speech) is not impacted if background noise is added or eliminated. Invariances impact the selection of a similarity function, especially if the similarity definition is based on a different set of invariances than the underlying features.
  - **Normalization:** a common problem of data manipulation is the need to normalize value ranges before combining them. For instance, if we deal with 10-dimensional feature vectors and use an Euclidean distance to describe similarity, the ranges of all dimensions should be normalized to allow for such a combined distance measure. Otherwise, the dimension with the large range will dominate the ones with small ranges. Normalization also encompasses dimensionality reduction and correlation analysis. Assume again the 10-dimensional feature vector: if several dimensions strongly correlate, the Euclidean distance grows faster for changes of these correlated values (the difference becomes replicated in multiple dimensions) than in uncorrelated dimensions. Dimensionality reduction (Principal Component Analysis) eliminates correlation. Alternatively, a special quadratic function can be used to adjust for the correlation.

- A very common method to measure similarity is through a distance function. Assume we have a feature space $\mathbb{R}^d$ with d dimensions. A query Q is mapped into this feature space yielding a feature vector $\boldsymbol{q} \in \mathbb{R}^d$. The same mapping leads to feature vectors $\boldsymbol{p}_i \in \mathbb{R}^d$ for each of the media objects P_i. In case of uncorrelated dimensions, a weighted L_k-norm is a good selection to measure distances
  - The weights are chosen such that the ranges of all dimensions become comparable. Several strategies exist to compute the weights. Here are two examples:

$$w_j = \frac{1}{\max_i p_{i,j} - \min_i p_{i,j}}$$

$$w_j = \frac{1}{\sigma_j}$$ with σ_j being the standard deviation of values in dimension j

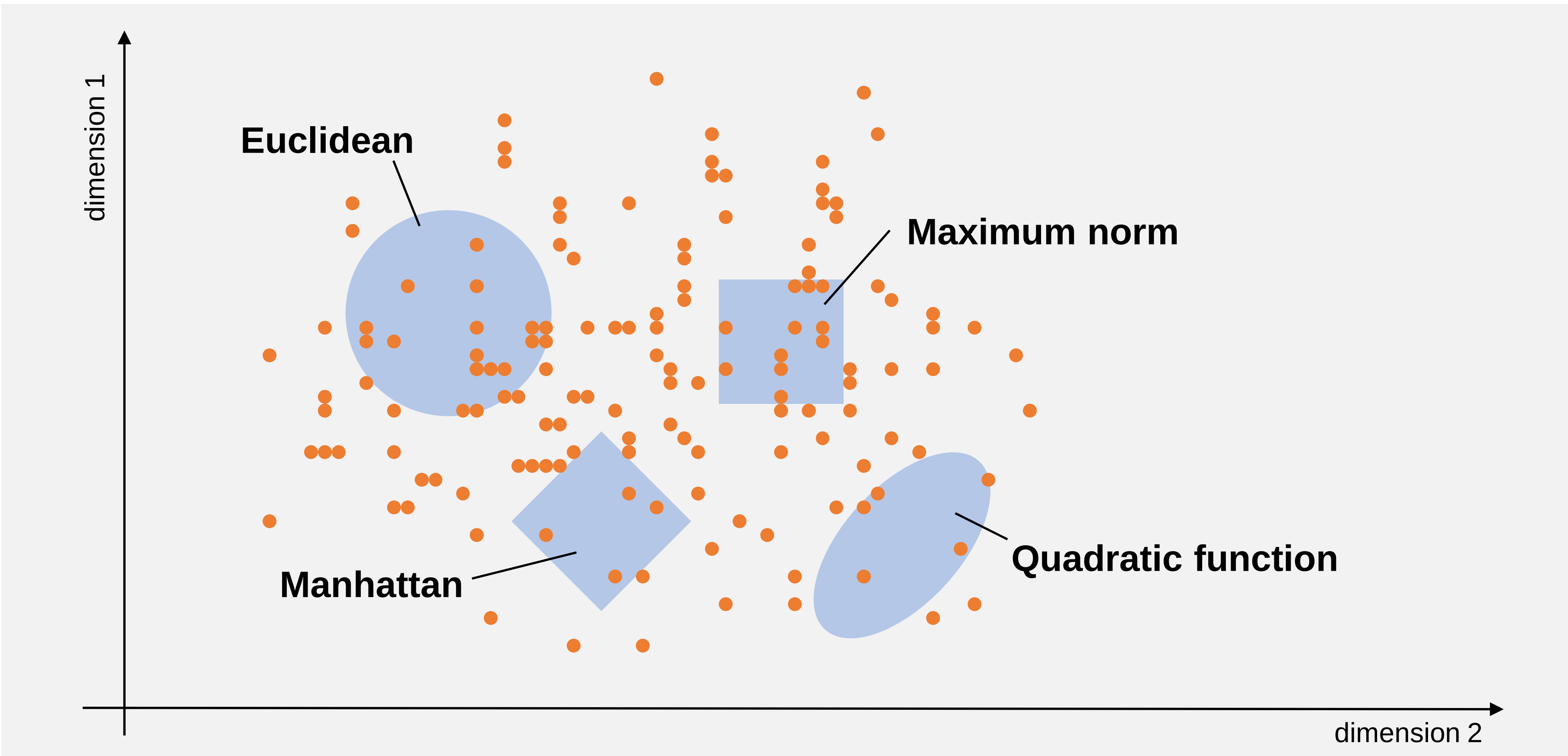
  - The distance between the query vector $\boldsymbol{q}$ and media vector $\boldsymbol{p}_i$ is then:
    - L_1-norm or Manhattan distance: $$\delta(\boldsymbol{q}, \boldsymbol{p}_i) = \sum_j w_j \cdot |q_j - p_{i,j}|$$
    - L_2-norm or Euclidean Distance: $$\delta(\boldsymbol{q}, \boldsymbol{p}_i) = \sqrt{\sum_j w_j^2 \cdot (q_j - p_{i,j})^2}$$
    - L_k-norm or k-norm: $$\delta(\boldsymbol{q}, \boldsymbol{p}_i) = \sqrt[k]{\sum_j w_j^k \cdot (q_j - p_{i,j})^k}$$
    - L_∞-norm or Maximum norm: $$\delta(\boldsymbol{q}, \boldsymbol{p}_i) = \max_j \left(w_j \cdot |q_j - p_{i,j}|\right)$$

– For correlated dimensions, we can use a quadratic function with a matrix $\mathbf{A} \in \mathbb{R}^d$ that compensates correlation. In this case, weights are already factored into the correlation matrix:

- Quadratic function:

$$\delta(\boldsymbol{q}, \boldsymbol{p}_i) = (\boldsymbol{q} - \boldsymbol{p}_i)^\top \mathbf{A} (\boldsymbol{q} - \boldsymbol{p}_i)$$

– The following visualization shows all distance measures. The blue area depicts the neighborhood areas around the centers of the areas (e.g., a query vector):



- Example for weights: consider the following two dimensions
  - In dimension d_1, all values are between 0 and 1.
  - In dimension d_2, all values are between 100 and 200.

  If we would apply an unweighted distance function, dimension d_2 would dominate dimension d_1. In other words, regardless of how close the features are in dimension d_1, only the difference in dimension d_2 really matters. Similarity is hence based (almost) entirely on dimension d_2. With the weights, we can normalize the different ranges along dimensions. Note that all metrics are based on differences so that the absolute values do not matter if ranges are similar.

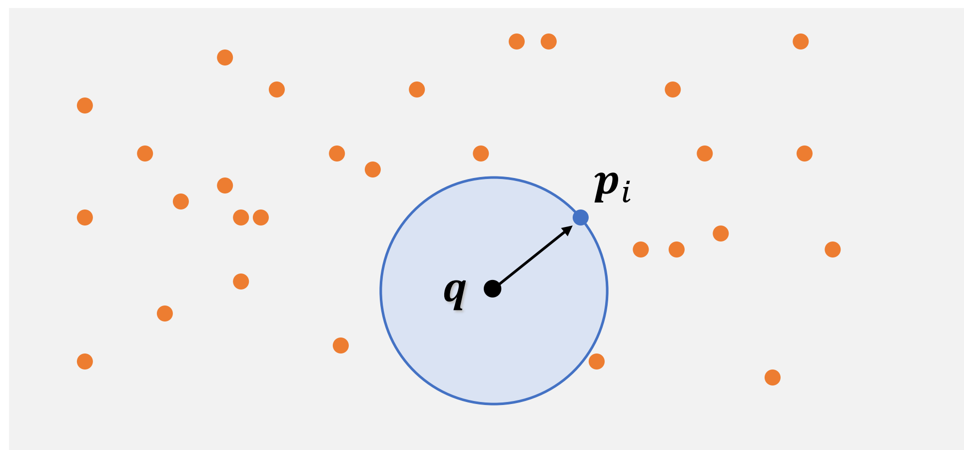
- Searching for the most similar object translates to a search for the object with the smallest distance, the so-called **nearest neighbor**. We note the reversed relationship between similarity values and distances:
  - large distances correspond to low similarity values
  - small distances correspond to high similarity values

  We can express similarity search as a nearest neighbor search:

Nearest Neighbor Problem:

- Given a vector $\boldsymbol{q}$ and a set $\mathbb{P}$ of vectors $\boldsymbol{p}_i$ and a distance function $\delta(\boldsymbol{q}, \boldsymbol{p}_i)$
- Find $\boldsymbol{p}_i \in \mathbb{P}$ such that:

$$\forall j, \boldsymbol{p}_j \in \mathbb{P}: \delta(\boldsymbol{q}, \boldsymbol{p}_i) \le \delta(\boldsymbol{q}, \boldsymbol{p}_j)$$

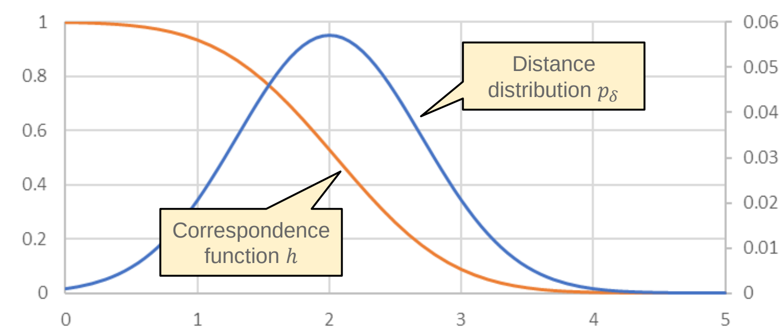


- If we want to obtain similarity values from the distances, we need a so-called **correspondence function** h. Let $\sigma(\boldsymbol{q},\boldsymbol{p}_i)$ denote a similarity function between query vector $\boldsymbol{q}$ and a media vector $\boldsymbol{p}_i$. The following properties must hold:
  - $\sigma(\boldsymbol{q},\boldsymbol{p}_i)$ is in the range $[0,1]$
  - $\sigma(\boldsymbol{q},\boldsymbol{p}_i)=0$ denotes total dissimilarity between query vector $\boldsymbol{q}$ and a media vector $\boldsymbol{p}_i$
  - $\sigma(\boldsymbol{q},\boldsymbol{p}_i)=1$ denotes maximum similarity between query vector $\boldsymbol{q}$ and a media vector $\boldsymbol{p}_i$
  - The correspondence function translates between distances and similarity values as follows

$$\sigma(\boldsymbol{q},\boldsymbol{p}_i)=h(\delta(\boldsymbol{q},\boldsymbol{p}_i)) \qquad \delta(\boldsymbol{q},\boldsymbol{p}_i)=h^{-1}(\sigma(\boldsymbol{q},\boldsymbol{p}_i))$$

    It must fulfil the following constraints
    - $h(0)=1$
    - $h(\infty)=0$
    - $h'(x)\leq 0$ (h must be a decreasing function)
  - The best method to build a correspondence function is to use the distance distribution p_δ. We obtain the mapping by integrating the distribution function up to the given distance and subtract that value from 1. This guarantees that all constraints hold true:

$$h(x)=1-\int_0^x p_\delta(x)dx$$



4.3 Metadata Extraction

- There is a simple way to close the semantic gap: we annotate the media files with keywords and derive higher-level semantic features similar to the techniques we have seen in text and web retrieval. In this context, the meta data is a low-level feature in the form of structured or unstructured text, while the terms extracted and the reasoning on the terms denote the higher level features (which are not inferred directly from the raw signal).
- However, it costs about $50 to $100 to annotate an image with the necessary level of detail and quality. With the billions of images and the limited revenue generation from such annotations, this clearly is not an attractive path. Or would you pay $100'000 for the 1'000 photos from your last vacation? Clearly not. So we need a cleverer approach to automate annotations as much as possible. This is not always feasible.
- We can divide meta data roughly into two groups:

Technical Metadata

- Administrative Data
- Media Properties
- Creation Information

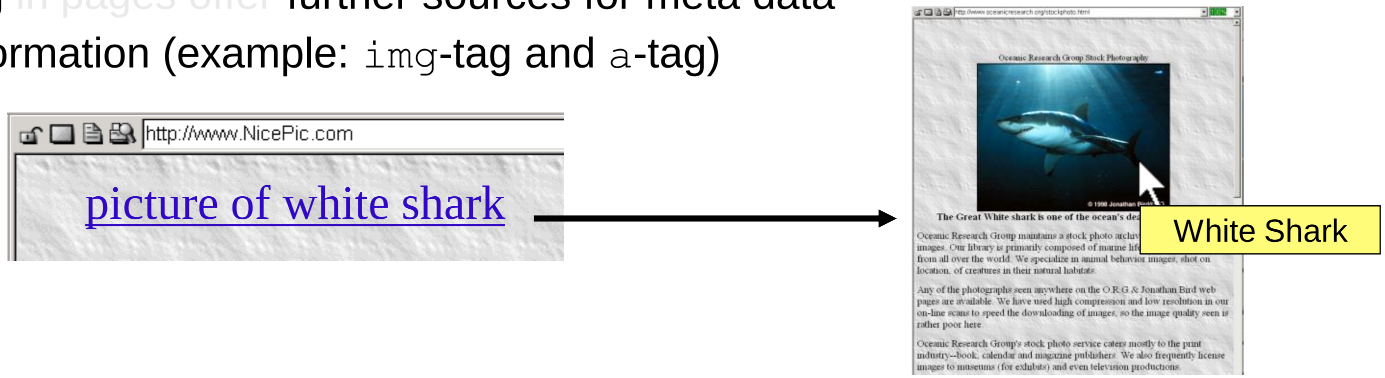
Subject Metadata

- Title, Captions
- Descriptions
- Relations

- There are **many standards for metadata** description like **RDF, Dublin Core, Dublin Core Metadata** Initiative and others that define standards how to annotate media files. They all are part of the **semantic web initiatives** to provide better connection of information. In the context of web pages, the `meta`-tag in the header holds all meta information about the current web page. Its format is: `<meta name="description" content="text">`. Next to description, a number of **further meta data** items are possible:

name	content
description	short description of web page
keywords	keywords associate with page
abstract	short narrative of content
author	author of this page
contact	contact person for this page
copyright	name of owner
dc.language	language of page (e.g., using RFC1766 and ISO 639)
dc.source	reference to page from which this page is derived
dc.creator	creator information for page
…12 more Dublin core tags and even more DCMI tags possible	

- In the context of multimedia content, the web offers more information than the simple meta information in the header section. Similar to what we have seen in web retrieval, links and embedding in pages offer further sources for meta data
  - Link information (example: `img`-tag and `a`-tag)



    - The `alt`-attribute in the `img`-tag is a good source for a caption. Sometimes the file name yields additional keywords of interest
    - Hypertexts annotate the referenced image (like we did for web pages) with additional keywords. These annotations contain keywords at different semantic levels. If an image is frequently referenced, we may find a complete description of the content from various perspectives and covering a wide range of user specific semantics.
  - A good source for keywords is the surrounding area on the web page. If we look before and after the image we find title, caption, and relevant keywords for the image. The same applies to links (also within the same page) to media objects. The surrounding area holds many interesting aspects.
    - What means surrounding? and how far does it stretch? This may also lead to false annotations

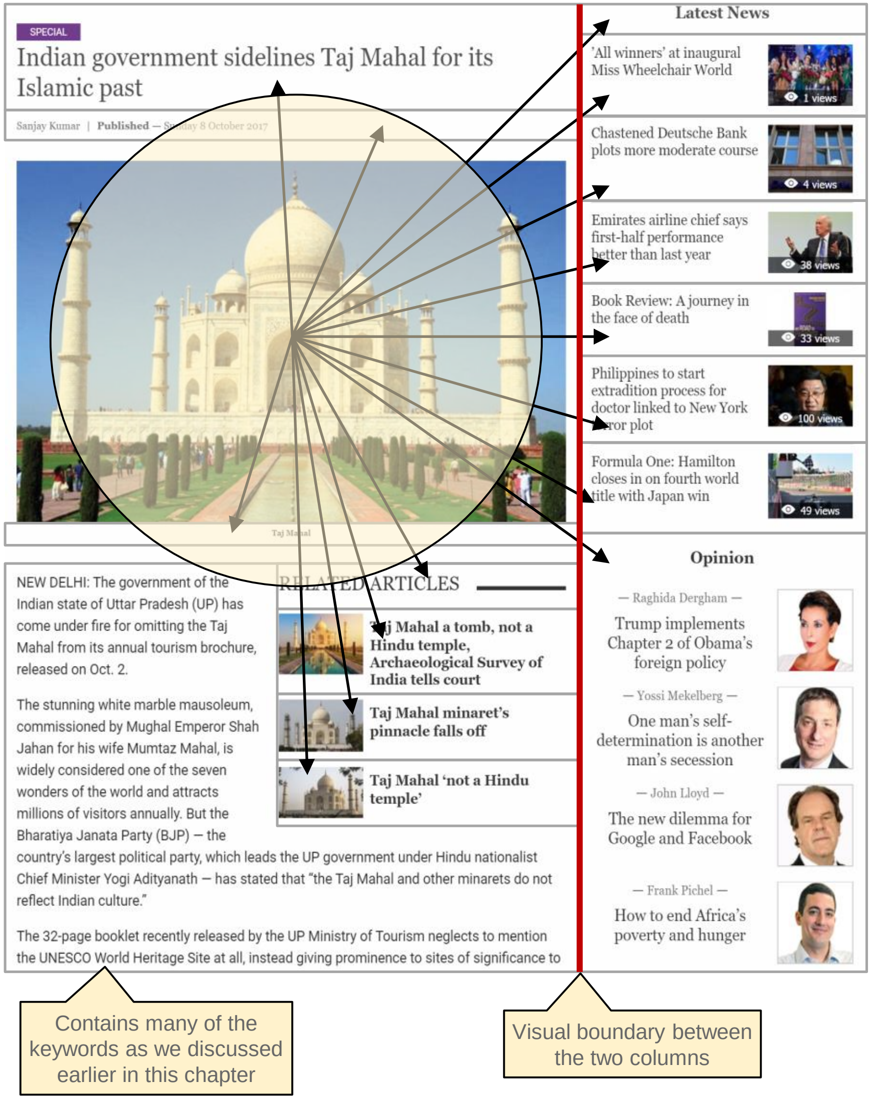
- Extracting information from the web page (basics)
  - The meta information of the web page is a good source for descriptors of an embedded image. In addition, headings or table headers before the image may contain further relevant information. The larger the document, the less likely such association may hold true
  - The window (in terms of characters in the HTML file) around the embedding holds many text pieces of potential relevance for the image. The size of the window must be carefully chosen to avoid wrong associations. Alternatively, we can weigh terms inversely to their distance to the embedding tag.

```
<HTML><HEAD>
   <TITLE>Linux is cool.</TITLE>
</HEAD>
<BODY BACKGROUND="./images/paper11.jpg">
<CENTER><H1>LINUX</H1>
<P>
<IMG SRC="./images/tux.gif"
   ALT="picture the penguin from linux">
<EM>This penguin, Tux, is the
           official mascot of Linux.</EM></CENTER>

<H2>MY FEELINGS ABOUT LINUX</H2>
I'll tell you, Linux has to be, ...
<P>
<H2>MY INVOLVEMENT WITH LINUX</H2>
...
</BODY>/HTML>
```

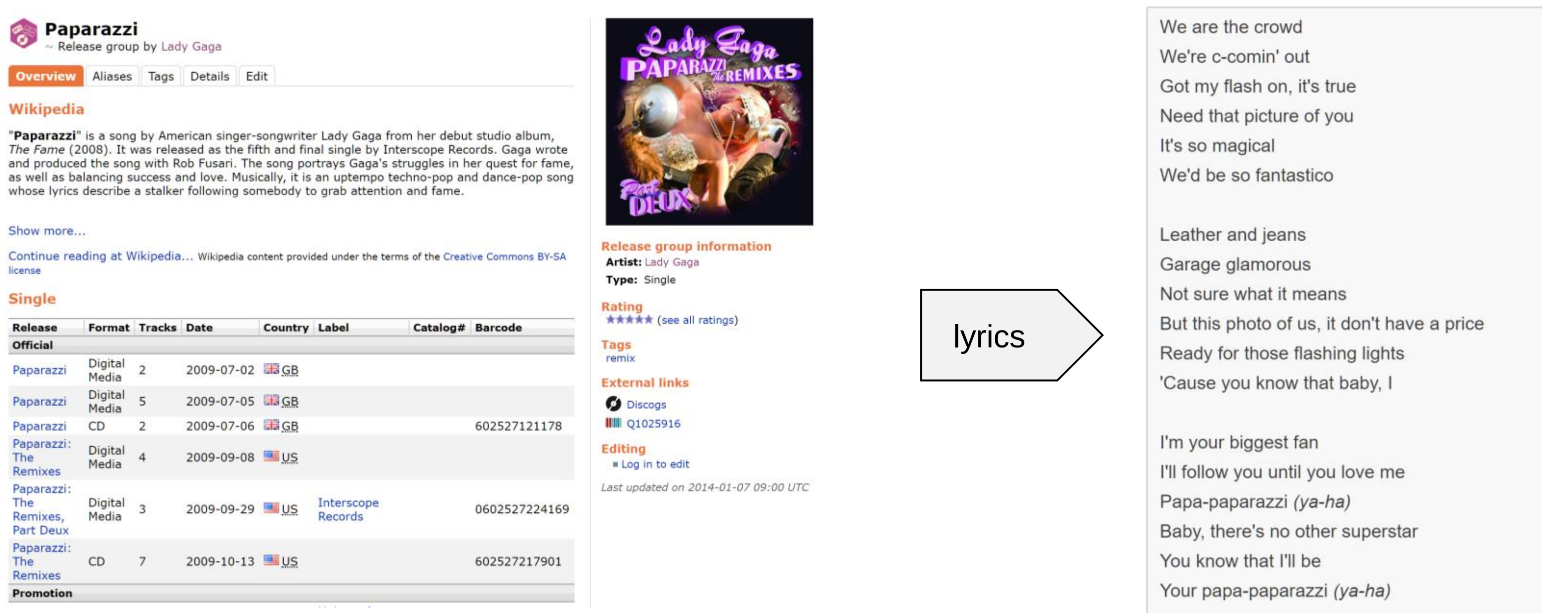
annotations

Source	Text
src-attribute	tux.gif
alt-attribute	picture the penguin from linux
title	Linux is cool.
h1	LINUX
em	This penguin, Tux, is the official mascot of Linux.
text	LINUX This penguin, Tux, is the official mascot of Linux. MY FEELINGS ABOUT LINUX

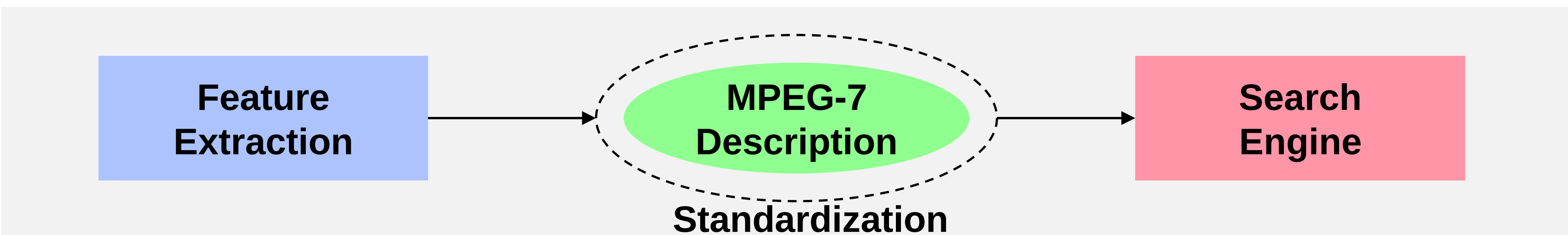


- An alternative approach uses visual closeness to annotate objects:
  - Instead of defining the neighborhood in the source code, it is defined by the proximity in the visual layout of the page (distance as perceived by reader)
  - Implementation:
    - Render the page and define core blocks on the page given the core structural elements (`div`, `p`, `table`, `form`, …)
    - Compute distances between these blocks and the embedded object. The distance can be any measure like pt or pixel.
    - Add penalties to the distance if there is a (visual) delimiter between the blocks. For instance, a line separating table cells. Column boundaries in a multi-column layout. Other blocks in between.
    - Define a neighborhood and add all blocks intersecting with that neighborhood. Use the distance as a weigh for the terms found within a block. Apply further weighting based on visual attributes such as bold, italic, header, …
    - Summarize descriptions with bag-of-words approach and associate it to the image.

- A more targeted approach is to “scrape” information on media objects, especially if they are highly standardized and categorized. With images, this is hardly achievable and only for sets of official catalogues. But for music and videos, this is the most common approach. Consider you want additional annotations for your music library to be able to find songs by keywords. A good starting point is MusicBrainz.org which catalogues a large portion of published songs and is entirely public domain (you can download the entire database).
  – Example below: for every song in a media library, we can extract information about the artist, about albums and releases, and about individual songs and interpretations of it. Using services like LyricWiki, we can obtain a full description of high-level semantics for our songs. If you combine several services, you can easily complete the descriptions of your media library.
  – Both IMDb and TMDb offer similar services for movies and series. TMDb is a community built database and free to use (with usage restrictions as per license agreement)



- **MPEG-7** is an ISO standard for multimedia content defined by the Motion Picture Expert Group in 2002. In contrast to MPEG-1, MPEG-2, and MPEG-4, the encoding format MPEG-7 is not about a new compression algorithm but focuses on meta information and its description
  - MPEG-7 defines a language to store meta information to
    - describe any multimedia document (images, audio files, video files)
    - describe possible descriptors and their relationships to each other
    - define descriptors
    - encode descriptors and prepare them for later indexing
  - The standard does not include:
    - the concrete implementations of feature extraction algorithms to not hinder development
    - filter and search algorithms to scan through MPEG-7 data
  - MPEG-7 bridges content provider and search engines with a standardized representation. It is the essential semantic glue between feature extraction and search engine. In the following, we look at the individual elements of the standard and how it fits into our model.



- Lets first consider how MPEG 7 stores technical meta data.

```
<MediaInformation>
    <MediaIdentification>
        <Identifier IdOrganization='MPEG' IdName='MPEG7ContentSet'>
            mpeg7_content:news1
        </Identifier>
    </MediaIdentification>

    <MediaProfile>

        <MediaFormat>
            <FileFormat>MPEG-1</FileFormat>
            <System>PAL</System>
            <Medium>CD</Medium>
            <Color>color</Color>
            <Sound>mono</Sound>
            <FileSize>666.478.608</FileSize>
            <Length>00:38</Length>
            <AudioChannels>1</AudioChannels>
            <AudioCoding>AC-3</AudioCoding>
        </MediaFormat>

        <MediaCoding>
            <FrameWidth>352</FrameWidth>
            <FrameHeight>288</FrameHeight>
            <FrameRate>25</FrameRate>
            <CompressionFormat>MPEG-1</CompressionFormat>
        </MediaCoding>

        <MediaInstance>
            <Locator>
                <MediaURL>file://D:/Mpeg7_17/news1.mpg</MediaURL>
            </Locator>
        </MediaInstance>

    </MediaProfile>

</MediaInformation>
```

Administrative Data

Media Properties

- Continuation of the technical meta data part:

```
<Creation>

    <Creator>
        <role>presenter</role>
        <Individual>
            <GivenName>Ana</GivenName>
            <FamilyName>Blanco</FamilyName>
        </Individual>
    </Creator>

    <CreationDate>
        1998-06-16
    </CreationDate>

    <CreationLocation>
        <PlaceName xml:lang="es">Piruli</PlaceName>
        <Country>es</Country>
        <AdministrativeUnit>Madrid</AdministrativeUnit>
    </CreationLocation>

    <Publisher xsi:type="Organization">
        <Name>TVE</Name>
        <ContactPerson> .... </ContactPerson>
    </Publisher>

</Creation>
```

Creation Information

- Now let us consider the subject meta data for the example:

```
<Title type="original">
    <TitleText xml:lang="es">
        Telediario (segunda edición)
    </TitleText>
    <TitleImage>
        <MediaURL>file://images/teledario_ori.jpg</MediaURL>
    </TitleImage>
```

Title, Captions

```
</Title>

<Title type="alternative">
    <TitleText xml:lang="en">
        Afternoon news
    </TitleText>
    <TitleImage>
        <MediaURL>file://images/teledario_en.jpg</MediaURL>
    </TitleImage>
```

Title, Captions

```
</Title>

<StructuredAnnotation>
    <Who>Fernado Morientes</Who>
    <WhatAction CSName='Sports'
        CSLocation='www.eurosport.xxx/cs/soccer/'> scoring  goal
    </WhatAction>
    <When>Spain Sweden soccer match</When>
    <TextAnnotation xml:lang='en-us'>
      This was the first goal of this match.
    </TextAnnotation>
```

Relations

```
</StructuredAnnotation>
```

- And the final part of subject meta data:

```
<Examples SemanticLabel="baldheaded man walking" Length="3"
        Confidence="1.0" DescriptorName="ColorHistogram">

    <Descriptor>
        4617 11986 938 2628 458 1463 5178 2258 444 134 69 456 9300 2810
        121 21 14 18 48 107 277 53 47 1926 8281 793 38 11 0 5 201 28 0
        1 1 2 23 252 122 6 3 433 1517 46 1 1 0 0 0 0 0 0 0 0 2 55 13560
        3326 678 221 1610 5602 916 32 8 1 21 58 11 1 0 0 2 61 331 179
        14 7 2388 6213 51 0 0 0 0 0 0 0 0 0 0 2 337 243 0 0 220 194 0 0
        0 0 0 0 0 0 0 0 0 0 383 3172 1072 51 20 91 128 0 0 0 0 0 2 4 0
        0 0 0 89 757 694 0 0 217 39 0 0 0 0 0 0 0 0 0 0 0 0 912 210 0 0
        0 0 0 0 0 0 0 0 0 0 0 0 0 0 55
    </Descriptor>

    <Descriptor>
        1764 18807 725 816 553 1784 7133 1325 81 3 8 110 5621 2323 34
        11 0 3 12 82 156 26 11 700 3060 63 7 0 0 0 1 0 0 1 0 0 16 95 40
        4 0 16 20 1 0 0 0 0 0 0 0 0 0 0 0 17 13534 3211 523 126 1123
        5181 347 37 0 0 0 5 8 2 1 0 2 17 261 168 3 0 997 2635 3 0 0 0 0
        0 0 0 0 0 0 2 292 39 0 0 17 1 0 0 0 0 0 0 0 0 0 0 0 0 157 861
        430 3 0 26 14 0 0 0 0 0 0 0 0 0 0 0 21 608 215 0 0 81 1 0 0 0 0
        0 0 0 0 0 0 0 0 373 37 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 9
    </Descriptor>

    <Descriptor>
        9742 15760 1455 2216 475 1356 4771 2328 714 329 193 420 6954
        6087 298 15 15 22 35 119 74 115 24 1253 7629 352 14 5 1 3 85 99
        0 0 0 0 0 11 0 6 0 335 717 9 0 0 0 0 0 0 0 0 0 0 0 0 12332 3066
        991 157 1048 4836 469 14 1 0 0 160 80 4 0 0 0 13 217 101 53 0
        3450 6079 12 0 0 0 0 0 0 0 0 0 0 0 6 0 0 0 338 64 0 0 0 0 0 0 0
        0 0 0 0 0 0 2439 718 15 0 81 41 0 0 0 0 0 0 0 0 0 0 0 0 65 0 0
        0 447 43 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
        0 0 0 0
    </Descriptor>

</Examples>
```

Descriptions

Descriptions

Descriptions

- Discussion: a good summary of the challenges around meta data is given by Cory Doctorow which he calls the seven insurmountable obstacles between the world as we know it and meta-utopia:
  - **People lie**. Metadata cannot be trusted because there are many unscrupulous content creators who publish misleading or dishonest metadata in order to draw traffic to their sites.
  - **People are lazy**. Most content publishers are not sufficiently motivated to carefully annotate all the content that they publish.
  - **People are stupid**. Most content publishers are not intelligent enough to effectively catalog the content they produce.
  - **Mission impossible—know thyself**. Metadata on the web cannot be trusted because there are many content creators who inadvertently publish misleading metadata.
  - **Schemas aren't neutral**. Classification schemes are subjective.
  - **Metrics influence results**. Competing metadata standards bodies will never agree.
  - **There's more than one way to describe something**. Resource description is subjective.

- Do we ignore meta data, then? Of course not, but we need to be careful what we are doing with the information provided. After all, a lot of the meta data can be extremely useful if the quality is right (see for instance MusicBrainz.org).
  - Observational meta data (automatically generated while crawling the web) is useful if it is hard to game the system (see PageRank as a good example).
  - Need to take the trustworthiness of the data provider into account. Google did so by trusting the users that link to a page more than the author of that page.

4.4 Features for Images

- We first look at low-level feature extraction from images based on the raw signal information. The process is divided into four steps:

Step 1: **Image Normalization** → Step 2: **Image Segmentation** → Step 3: **Feature Extraction** → Step 4: **Feature Aggregation**

  - **Image Normalization** depends on the data sets and includes a number of pre-processing steps including noise elimination, normalization of signal information, adjustments and corrections of the raw data. For example, when analyzing frames in an interlaced video sequence, deinterlacing is a typical step to reduce combing effects that interfere with feature extraction
  - **Image Segmentation** partitions the image into sub-areas for which perceptual features are extracted. We distinguish between global features (for the entire image) and local features (for a region within the images). If we have local features, the aggregation step (4) is necessary to obtain a global feature for the image.
  - **Feature Extraction** describes the signal information based on perceptual aspects such as color, texture, shape, and points of interest. For each category, a number of methods exists with different invariances (e.g., robustness against scaling, translation, rotation). We do not consider labeling of images in this chapter (see the next chapter for high-level features)
  - **Feature Aggregation** summarizes perceptual features to construct a final descriptor (or a set of descriptors). The aggregation often uses statistical approaches like mean values, variances, covariances, histograms, and distribution functions. With local features, we can further derive statistical measure across the regions (e.g., self-similarity, mean values, variances, covariances). In the following we often discuss feature aggregation together with the feature extraction method.

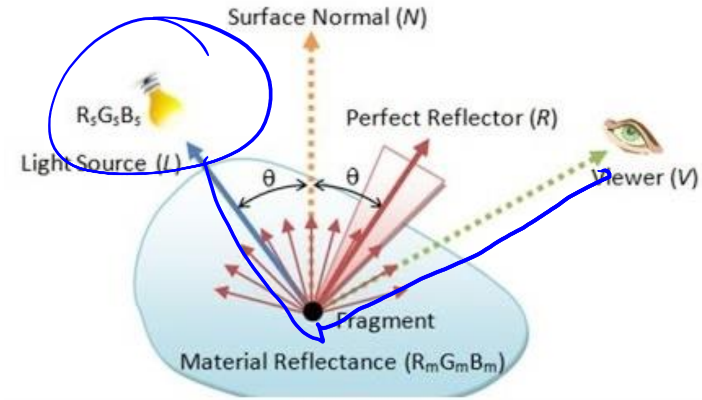
- **Feature Design:** before we design features, we need to define the desired invariance properties of the feature. For instance:
  - Translation invariant: (small) shifts of the picture have no significant impact on feature values
  - Rotation invariant: rotations of the image have no significant impact on feature values
  - Scale invariant: up- or down-sampling does not change the feature value. Note that scale differences are very common due to different image resolutions. In the absence of a normal sized scale, it is even more important to demand scale invariance
  - Lightning invariant: Adjustments of lightning (daylight, artificial light, brightness adjustments, gamma corrections) have no significant impact on feature values
  - Noise robustness: noise, JPEG artefacts, quantization errors, or limited color gamut have no significant impact on feature values

  We already have discussed global vs local features as a further invariance constraint.

4.4.1 Visual Perception and Processing

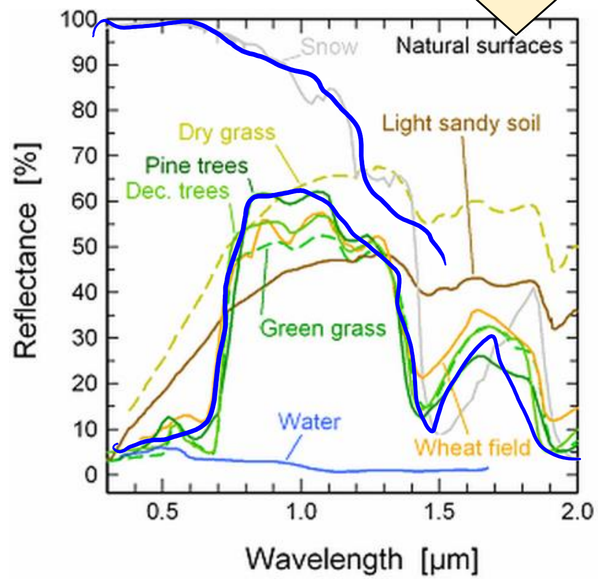
- Let's first consider how we perceive and process visual information. Perception of light is the result of illumination of an object and the amount of illumination that is reflected by the objects in front of us:
  - **Illumination** $l(x,y,z)$ is the amount of lumens per square meter (=lux). Lumen is a measure of energy per second modelled along the eye's sensitivity range of light.
  - **Reflectance** $r(x,y,z)$ is the amount of illumination reflected by the surface of objects. Reflectance is a function of wavelength, absorption, and direction of illumination.

  Typical illuminance and reflectance values are given below:



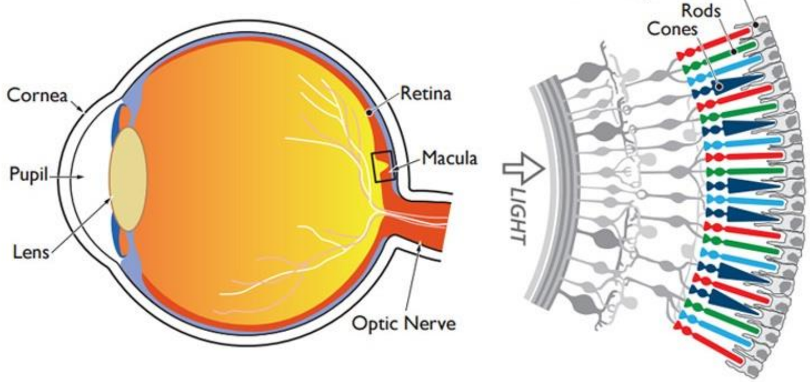
Illuminance (lux)	Surfaces illuminated by
0.0001	Moonless, overcast night sky
0.05–0.36	Full moon on a clear night
20–50	Public areas with dark surroundings
50	Family living room lights
100	Very dark overcast day
320–500	Office lighting
400	Sunrise or sunset on a clear day.
1000	Overcast day; typical TV studio lighting
10,000–25,000	Full daylight (not direct sun)
32,000–100,000	Direct sunlight

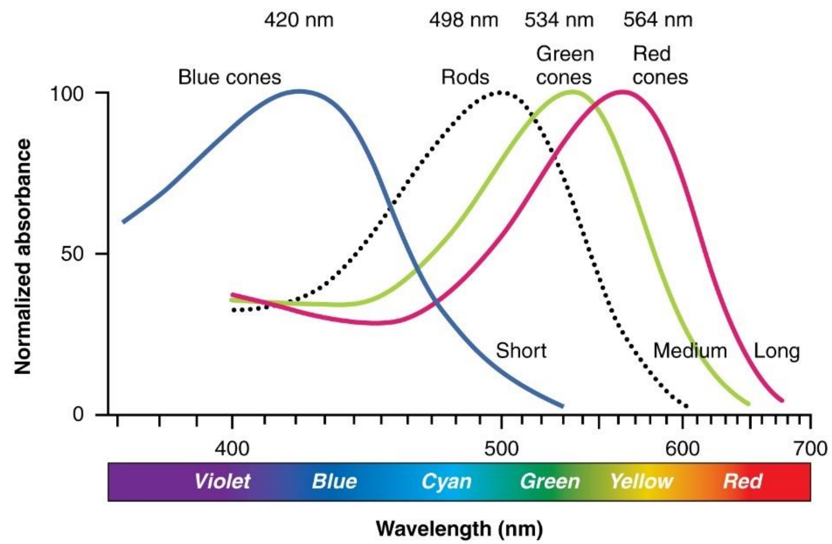
Chlorophyll has its reception peaks in the blue and red spectrum of light. Hence, we observe only the reflected green spectrum of light.



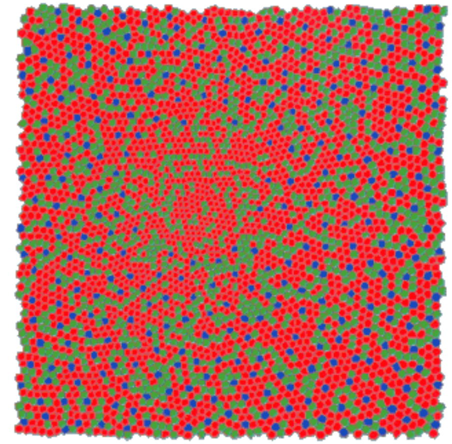
- The eye receives light and translates the wavelengths into electro-chemical impulses
  - The cornea, pupil, and lens form an adaptive optical system to focus on objects (distance) and adjust to light exposure (aperture). The lens works like an ordinary camera and projects an (upside-down) image of the world onto the retina at the back side of the eye.
  - The retina consists of three cone types and rods; they are the photoreceptors that transform incoming light energy into neural impulses. The cones enable color vision, specialize on different wavelength ranges, and are very frequent in the center of vision (macula and fovea)
    - L-cone (long wavelength) peak at 564nm corresponding to the color red
    - M-cone (medium wavelength) peak at 534nm corresponding to the color green
    - S-cone (short wavelength) peak at 420nm corresponding to color blue

    The rods perform better at dimmer light and are located at the periphery of the retina. They focus on peripheral vision and night vision.





- The human eye has about 6 million cones and 120 million rods. The distribution is roughly 1% S-cones (blue), 39% M-cones (green) and 60% L-cones (red). The picture on the right shows the distribution near the center of sight (blue cones occur here up to 7%). These ratios can greatly vary and cause color blindness. Cones are focused around the fovea (see lower right side), while rods fill the periphery of sight.

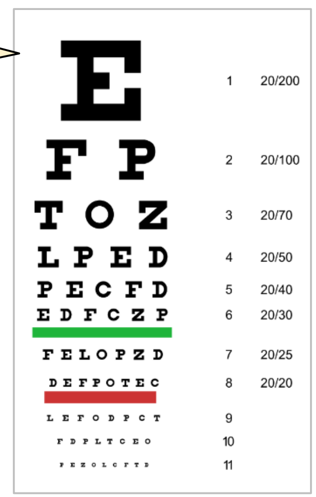


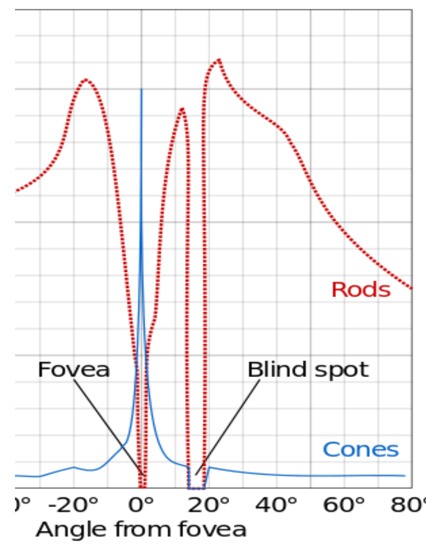
- **Visual Acuity** describes the clarity of vision and how well the eye can separate small structures. With the standard Snellen chart, a 20/20 vision denotes that the eye is able, at 20 feet distance, to separate structures that are 1.75mm apart. This corresponds to roughly one arcminute (1/60 degree). A 20/40 vision denotes that a person can see things at 20 feet distance as good as a normal person at 40 feet distance. The best observed vision for humans is 20/10. Visual acuity is limited by the optical system (and defects like short-sightedness) and the number of cones and rods per mm^2.

Ratio	Metric	Snellen	Arcminutes
2,0	6/3	20/10	0.5′
1,33	6/4,5	20/15	0.75′
1,0	6/6	20/20	1′
0,8	6/7,5	20/25	1.25′
0,67	6/9	20/30	1.5′
0,5	6/12	20/40	2′
0,4	6/15	20/50	2.5′
0,2	6/30	20/100	5′
0,1	6/60	20/200	10′
0,05	6/120	20/400	20′

1.4’ or less is required to drive a car

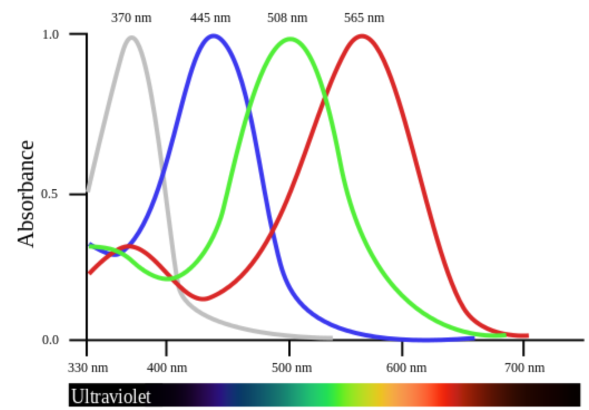
Standard Snellen Chart





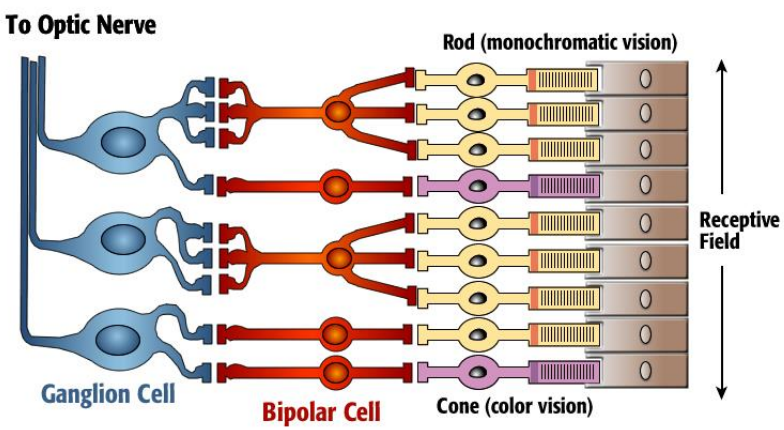
- The comparison with animals shows great differences in terms of visual sensing. A cat has a much lower visual acuity of 20/100 and less cone types (blue at 450nm and yellow at 550nm), but cats have better night vision (6-8 times) and a broader range of vision (200 degree vs 180 degree). Hence, a cat has a much blurred view compared to humans. Dogs are also dichromatic (blue/yellow) with a visual acuity of 20/75. Elephants have a 20/200 vision, rodents a 20/800 vision, bees a 20/1200 vision, and flies a 20/10800.

On the other side, eagles and bird of prey have a 20/4 vision (5 times better than the average human). In addition, some birds are tetrachromatic and see the word with four independent color channels. The goldfish and zebrafish also have four different cone types. The additional cone type is typically in the ultra-violet range with a peak at about 370nm.

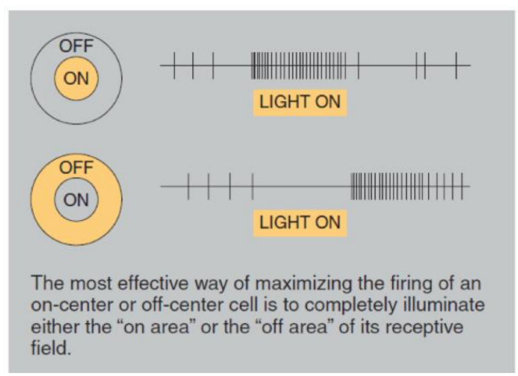


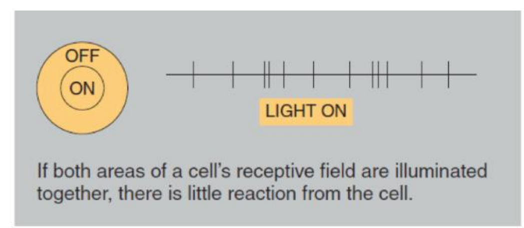
- **Conclusion:** our color vision is a sensation but not physics. To understand how we perceive images, we need to follow the way the human eye (and brain) processes light.

- The first processing starts within the retina (we will see similar concept in deep learning by means of convolution). The chemical process in the rods and cones release glutamate when its dark, and stop releasing glutamate when its light (this is unusual for a sensory system). The **Bipolar Cells** connect to several rods and cones (but never both together) and perform a simple operation:
  - On-Bipolar cells, fire when it is bright
  - Off-Bipolar cells, do not fire when it is bright
- The next stage, the **Ganglion Cells** build the first receptive fields combining various bipolar cells. In a nutshell, they perform edge detection with a center and a surround area.
  - On-Center ganglion fires, if center is bright and surrounding is dark
  - Off-Center ganglion fires, if center is dark and surrounding is bright
- Several additional cell types (horizontal cells, amacrine cells) act as inhibitors to accentuate contrast. This increased contrast can also lead to falsely under-/oversaturating dark/light boundaries. Lateral inhibition provides negative feedback to neighbor cells to further strengthen the contrast between strong and weak signals. This can lead to so-called after-images.



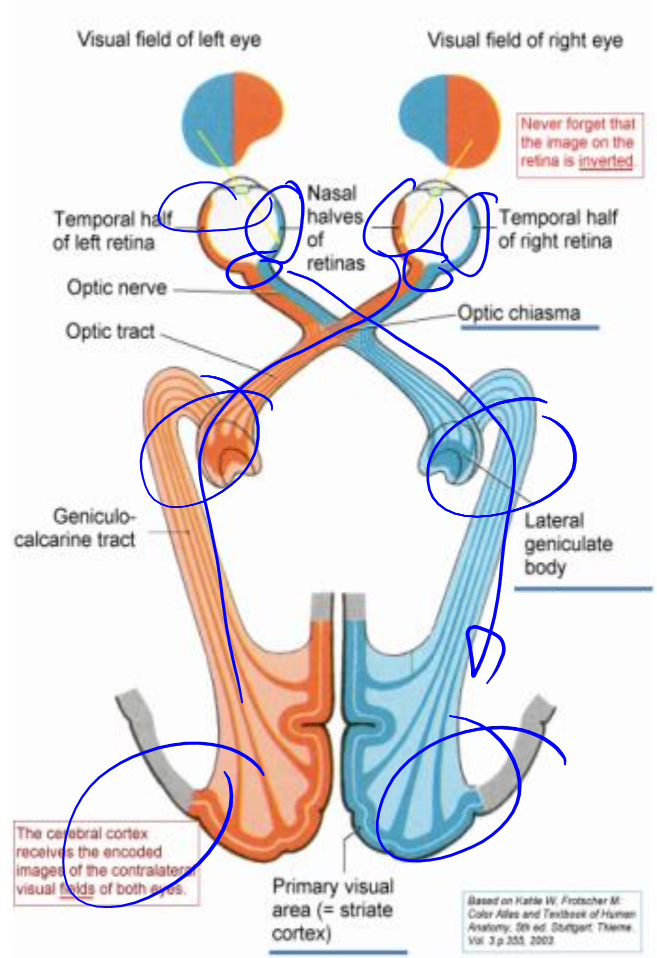
Bipolar cells can connect to many Ganglion Cells

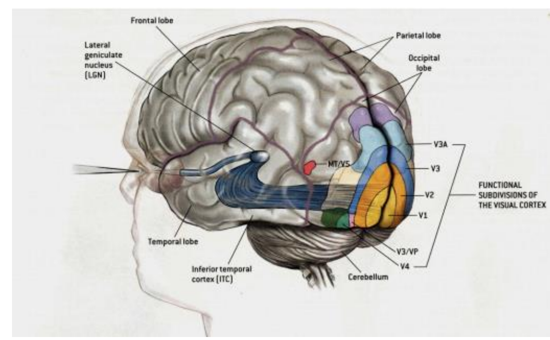




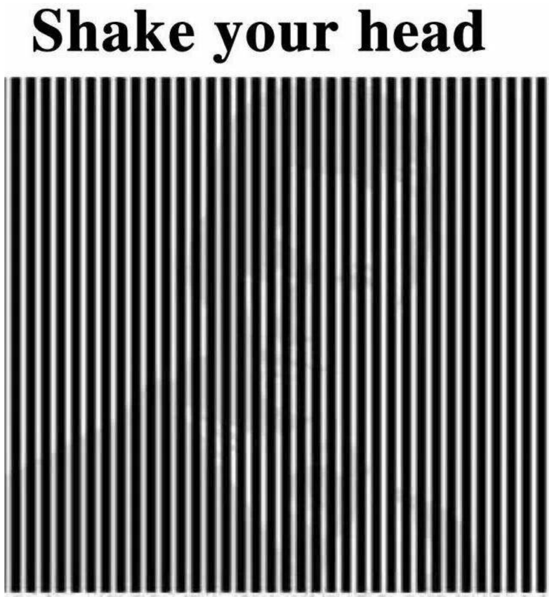
Different Ganglion Cells at work for their receptive field

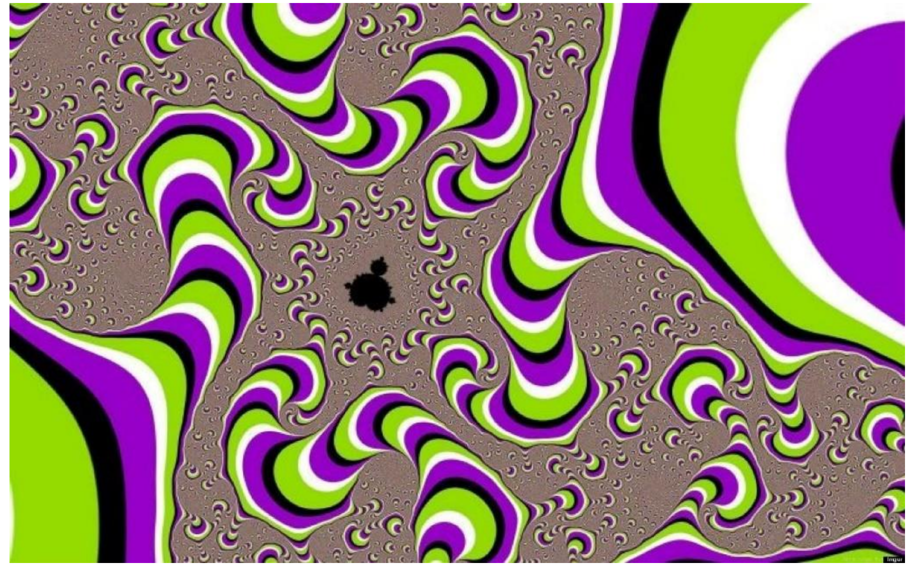
- The **Lateral Geniculate Nucleus (LGN)** performs similar receptive field functions as the ganglion cells but with massive feedback from the cortex. We first observe a split of the two visual fields (visual left is processed by the right side of the brain, visual right is processed by the left side). Secondly, the information of both eyes is combined. The first two layers focus on rods and the detection of movements and contrast. The next 4 layers process information from cones to perceive color and form (finer details).
- The **Primary Visual Cortex (V1)** performs detection of edges, orientation, some of them variant to position, others invariant to position. Neurons in the visual cortex fire when the defined patterns occur within their receptive fields. In the lower levels, the patterns are simpler; in higher levels, more complex patterns are used (e.g., to detect a face). The stream of information flows along two paths to higher levels.
  – The **Ventral Stream** (ventral=underside, belly) specializes on form recognition and object representation. It is connected with the long-term memory.
  – The **Dorsal Stream** (dorsal=topside, back) focuses on motion and object locations, and coordinates eyes, heads, and arms (e.g., reaching for an object)
- Cortical magnification denotes the fact that the majority of neurons act on the information in the center of vision (creating a much denser, magnified view of the center)

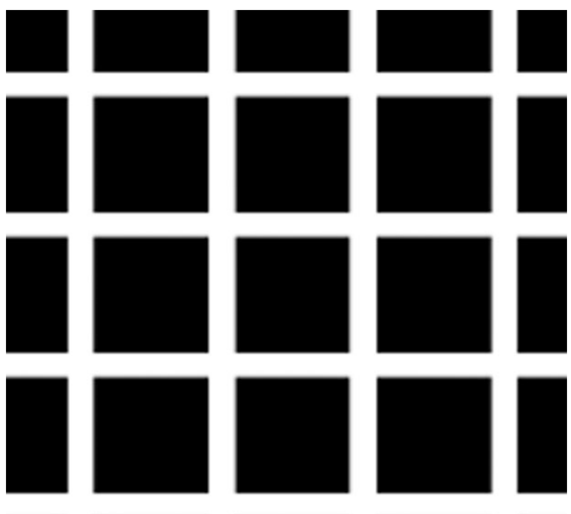


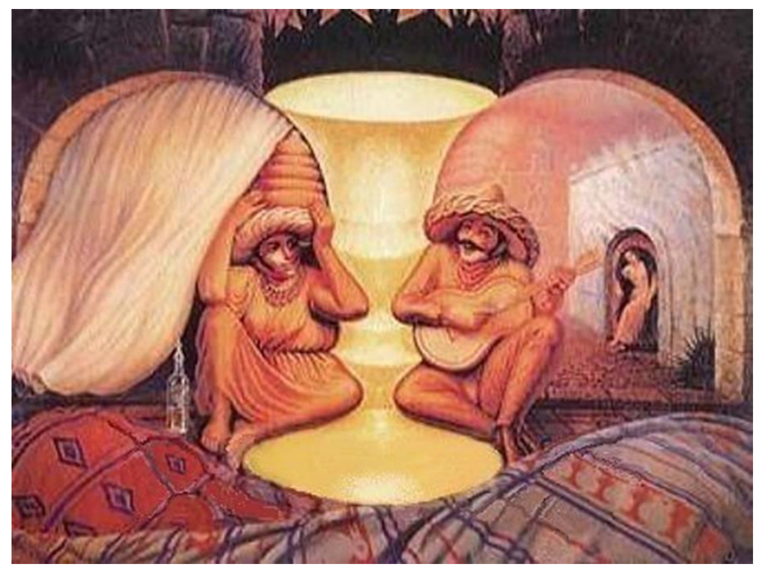


- The visual perception system is optimized for natural image recognition. Artificial illusions demonstrate very nicely how the brain processes the perceived environment in many ways:









4.4.2 Image Normalization (Step 1)

- In image processing, an image is usually described as a discrete function mapping a 2-dimensional coordinate to an intensity value (gray images) or a color value. We will use the function $i(x,y)$ and $\boldsymbol{i}(x,y)$ to denote such images:

grayscale images:	$i(x,y)\colon \mathbb{N}^2 \to [0,1]$
color images:	$\boldsymbol{i}(x,y)\colon \mathbb{N}^2 \to [0,1]^3 = \begin{bmatrix} r(x,y) \\ g(x,y) \\ b(x,y) \\ \alpha(x,y) \end{bmatrix}$
color channels (red)	$r(x,y)\colon \mathbb{N}^2 \to [0,1]$
color channels (green)	$g(x,y)\colon \mathbb{N}^2 \to [0,1]$
color channels (blue)	$b(x,y)\colon \mathbb{N}^2 \to [0,1]$
α-channel (transparency)	$\alpha(x,y)\colon \mathbb{N}^2 \to [0,1]$
with	$1 \le x \le N, 1 \le y \le M$

  – It is custom to start with the upper left pixel ($x = 1, y = 1$) and to end with the lower right pixel ($x = N, y = M$). x denotes the row in the image (vertical axis), while y denotes the column in the image (horizontal axis).

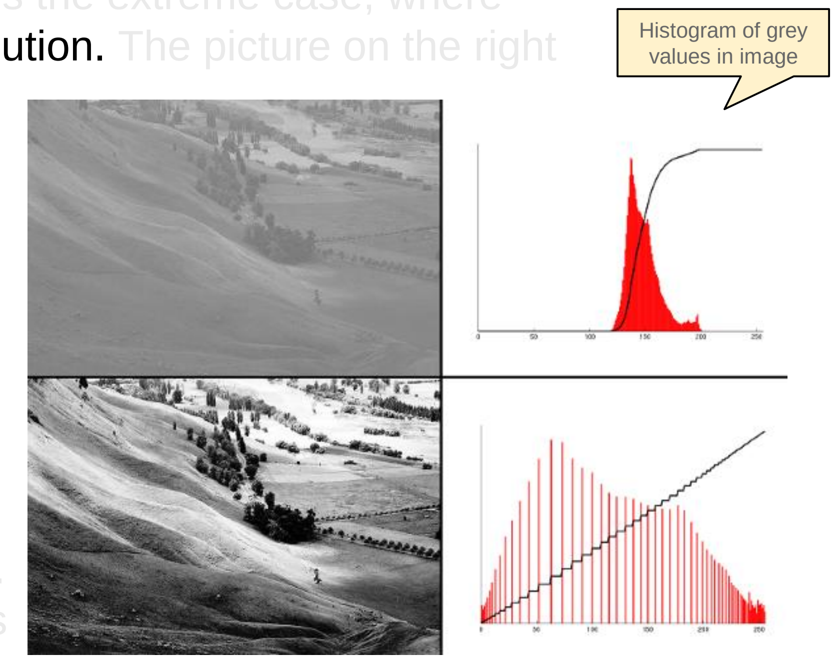
  – Quantization is often applied to avoid fixed point numbers in the image representation. Quantification is an approximation of the fixed point number as follows:

True Color (32-bit):	$\hat{f}(x,y)\colon \mathbb{N}^2 \to [0,255]$ approximating $f(x,y) = \frac{\hat{f}(x,y)}{255}$
Deep Color (64-bit):	$\hat{f}(x,y)\colon \mathbb{N}^2 \to [65535]$ approximating $f(x,y) = \frac{\hat{f}(x,y)}{65535}$

f denotes one of i, r, g, b, α

  – Other quantization with indexed colors exist but can be mapped to one of the above.

- Depending on the data collection, we need to perform a number of image processing steps to normalize the data sets and to achieve the best results when comparing features afterwards. Some of the processing steps ensure robustness against noise, rotation, color saturation, or brightness which are essential for the algorithms to work.
  - **Rotation** – if we need rotation invariant features (texture, shape) but do not have enough information to normalize direction, we can rotate the image in defined steps of degrees, extract features, keep all features for the image, but use them as individual representation (no combination of the features). A typical approach is by 90 degrees (which makes it simple). In object recognition (faces), more intermediate angles are possible (e.g., 15 degrees)
  - **Histogram normalization** – here, histogram means the distribution of brightness across the image. In poor sensing condition, the range of values can be very narrow, making it difficult to distinguish differences. Histogram equalization is the extreme case, where the range of values is forced to a uniform distribution. The picture on the right shows very nicely the increased contrast and the sharper contours of objects. With the original picture, edge detection may not lead to the expected results. Similar approaches are histogram shifts (lighter, darker), histogram spreading, or gamma correction.
  - **Grayscale transformation** – The original color image is transformed to a grayscale image. Depending on the source color model, different formulae define how to calculate the gray value. Often applied before texture and shape analysis as color information is not needed.



- **Scaling –** Up- or down-sampling of the image to fit within a defined range of acceptable sizes. For instance, a neural network might expect the input to fit into the input matrix. A shape or texture feature is sensitive to different scaling and may yield different results. The usual methods are bilinear or bicubic interpolation to avoid the creation of artefacts that could negatively impact the algorithms (in combination with Gaussian filters when down-sampling). If the algorithm is complex and expensive, down sampling is often applied to reduce the efforts. In such cases, the results are computed for the down-sampled image only, and then mapped back to the original image (see k-means clustering later on for image segmentation).
- **Affine Transformation –** The generalization of translation, rotation and scaling. The original coordinates (x, y) are mapped to a new pair (x', y') as follows:

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} a_{1,1} & a_{1,2} & a_{1,3} \\ a_{2,1} & a_{2,2} & a_{2,3} \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

  With this matrix representation, we can simplify the concatenation of various operators to obtain a single matrix again. To improve results, bilinear or bicubic interpolation is needed to estimate pixel values in the new matrix. Note: the affine transformation above does not necessarily map to a discrete and positive coordinate systems, and some areas in the new image space may have unknown values (think about a rotation by 45 degrees mapped to minimum bounding box).
- **Noise Reduction / Sensor Adjustments –** Sensors, transcoding and digitization can add noise (think of white and black pixels across the image) that can significantly impact the feature extraction process. Common methods are mean filter or Gaussian filters as described next. Other adjustments may include color corrections, distortions, moiré patterns or compression artifacts.

– **Convolution** is a mathematical operation that combines two functions to produce a new function. It is similar to the cross-correlation but considers values "backwards" and integrates them. The discrete two-dimensional form is given as ($*$ denotes the convolution operation)

3x3

$$(f * g)[x,y] = \sum_{n=-\infty}^{\infty} \sum_{m=-\infty}^{\infty} f[x-n][y-m] \cdot g[n][m]$$

Image → Kernel

- In image processing, g is called the Kernel and is typically a very small two-dimensional quadratic (and often symmetric) function with range $[-K,K] \times [-K,K]$ with small values $K = 1,2,3,4,\ldots$. Applied to an image channel $f(x,y)$ we obtain

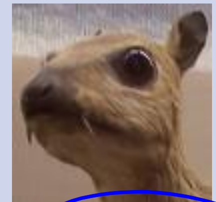
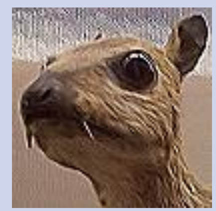
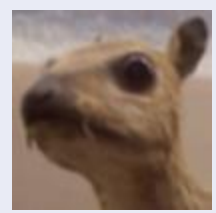
$$(f * g)[x,y] = \sum_{n=-K}^{K} \sum_{m=-K}^{K} f[x-n][y-m] \cdot g[n][m]$$

- As a visualization, assume we calculate the convolution of a 3x3 image with a 3x3 kernel for the center point of the image ($x = y = 2$). For example:

$$\left(\begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} * \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}\right)[2,2] = (i \cdot 1) + (h \cdot 2) + (g \cdot 3) + (f \cdot 4) + (e \cdot 5) + (d \cdot 6) + (c \cdot 7) + (b \cdot 8) + (a \cdot 9)$$

Note that the Kernel is actually flipped horizontally and vertically and then dot-wise multiplied with each image element. If the Kernel is symmetric, we can just apply the dot-wise multiplication to compute the convolution. Further note, that the Kernel is moved with its center across the image to compute a new value for that current pixel. If the Kernel overlaps the image, we use 0-padding for pixels beyond the boundary to keep image dimensions.

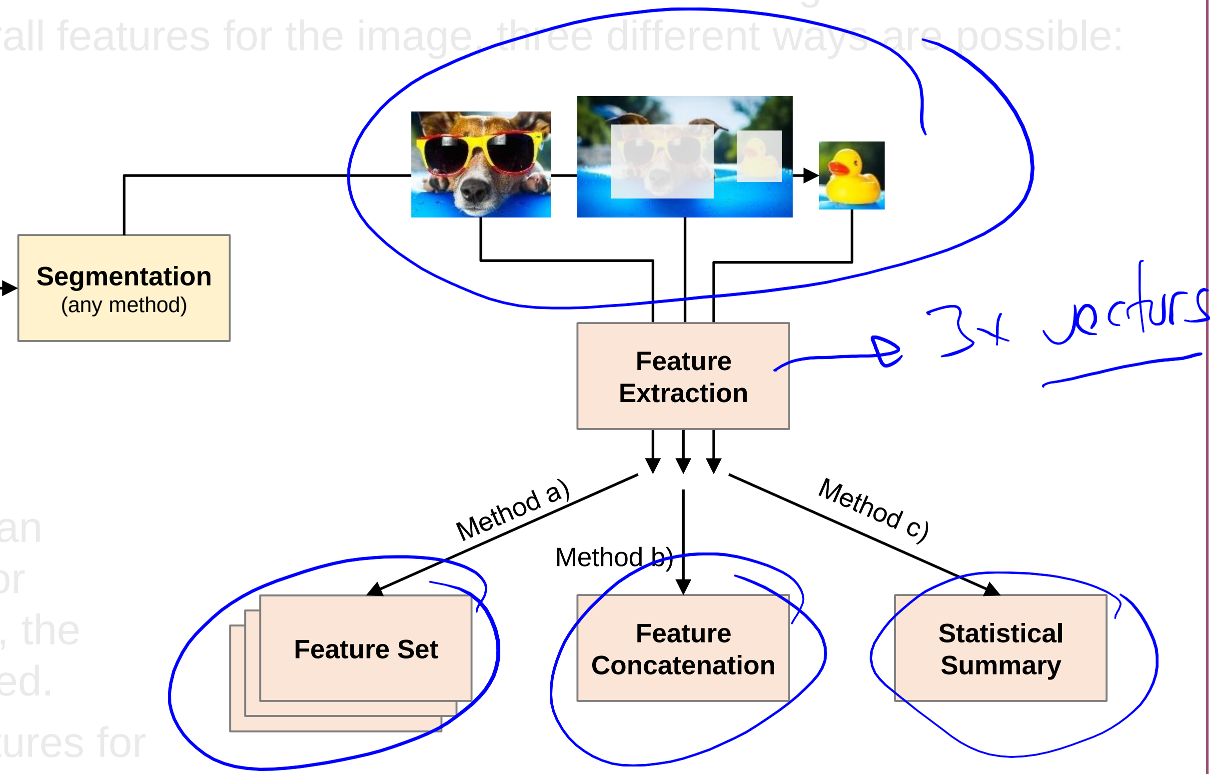
- Kernel Examples: (taken from Wikipedia for illustration purposes). When defining a Kernel, it is important to normalize the output by the sum of all Kernel values, otherwise channel values may exceed the defined boundaries ([0,1] or, if quantized, [0,255]).

Operation	Kernel	Image Result
Identity	$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	
Edge Detection	$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$	
Sharpen	$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$	
Box Blur	$\frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$	

Here, we need to divide by the sum of the Kernel values. In all other examples, that sum is 1.

4.4.3 Image Segmentation (Step 2)

- Feature design may include the capturing of location information (much like we did with position information in text retrieval). Segmentation define areas of interest within the image for which the features are computed. To obtain overall features for the image, three different ways are possible:



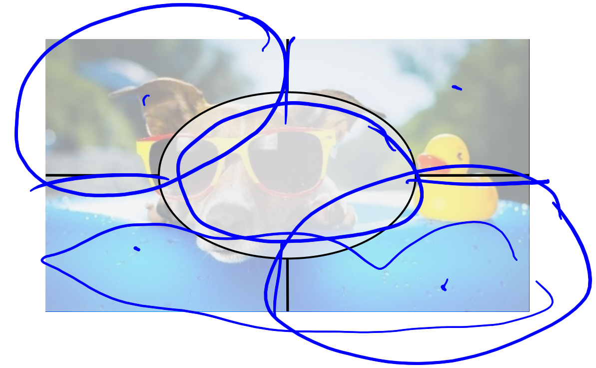
a) **Feature Sets** – for each segment an individual feature is stored. If one or more feature match with the query, the image (with the segment) is returned.

b) **Feature Concatenation** – the features for each segment are combined to form an overall feature for the image. This approach is only meaningful for pre-defined segmentations but not for object related segmentation with varying number of segments.

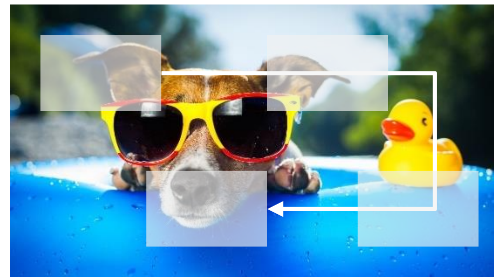
c) **Statistical Summary** – the features are summarized with statistical operators like mean, variance, co-variance, or distribution functions. The statistical parameters describe the image.

If the segmentation only yields one segment (global features), all methods become identical.

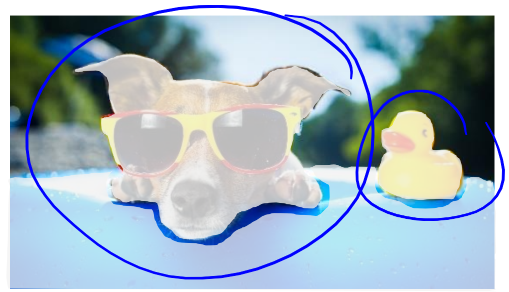
- We can segment images with three approaches (actually the first one does nothing)
  – **Global** features require the entire image as input. No segmentation occurs. This approach is often the standard in absence of a clear segmentation task. We will see later that with temporal media like audio and video, global features are very rare but quite common for still images.
  – **Static Segmentation** uses a pre-defined scheme to extract areas of interest from the image. There are two reasons for such a segmentation
    - Add coarse location information to the features. Typically, an image consists of a central area (the object) and four corner areas (as shown on the right). But any type of regular and potentially overlapping division is possible. Often, this method is combined with the concatenation of features to encode left/right, up/down, or center within the feature.



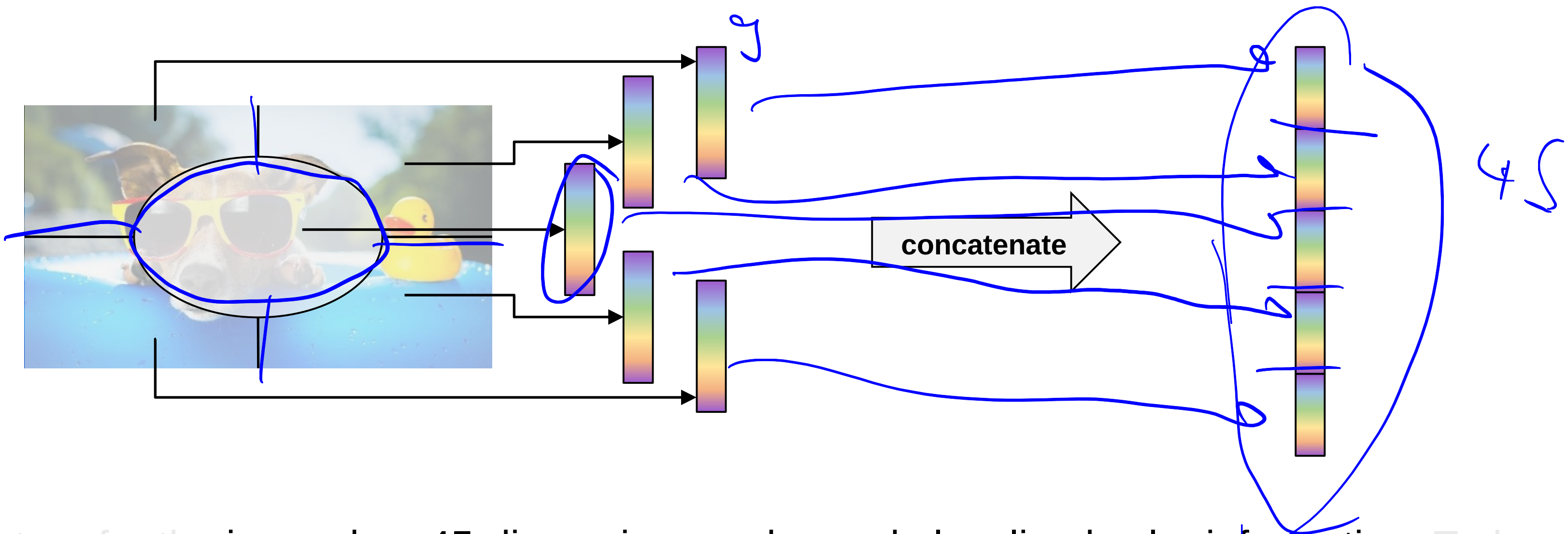
    - Process parts of the query image to detect similar features. We use a sliding window that moves from upper left to lower right in defined steps. For each position, features are extracted and used to find matches. For example, when detection faces the sliding window technique allows to find many faces together with their location from a given input picture (see next chapter).



  – **Object Segmentation** extracts areas with embedded objects in the picture (so-called blobs). These blobs are either analyzed individually or as a part of the image. Often, feature sets are used to enable individual retrieval of the blobs. We will study such an approach in the next chapter (k-means clustering).



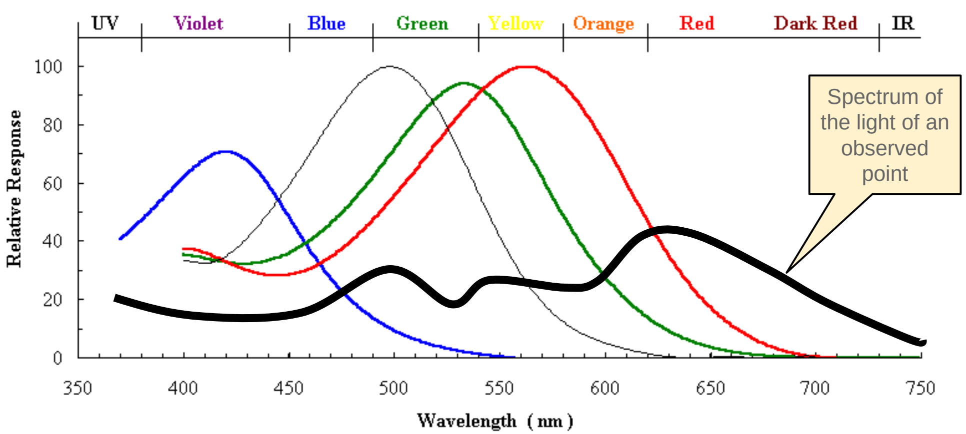
- Example: 9-dimensional color feature with 5 static segments
  – Segmentation creates 5 areas for each of which a 9-dimensional feature is extracted



  – The feature for the image has 45-dimensions and encode localized color information. To be similar with the above picture, the colors not only have to occur in a similar way but they also have to be in the same area. On the other side, we loose some invariances, like rotation. An upside-down version of the picture does not match with itself. On the other side, a blue lake does not match with the blue sky, a white background (snow) does not match with the white dress (center), and an object on the left does not match with the same object on the right.
  – We will see, that a single feature is often not sufficient to find similar pictures. Rather, we need to construct several (very similar) features to encode the different choices for variance and invariance. Segmentation, obviously, can both eliminate location information (for instance feature sets), enforce location (feature concatenation), or is liberal about the position (statistical summary and feature set).

4.4.4 Feature Extraction – Color Information (Step 3 & 4)

- We split the third step, feature extraction, into color, texture and shape information. We start with color in this subsection.
- Color perception is an approximation of the eye to describe the distribution of energy along the wavelength of electromagnetic signals. "Approximation" because the distribution cannot be described accurately with only 3 values, hence most information is lost. It is possible two construct two different spectra which are perceived exactly the same.

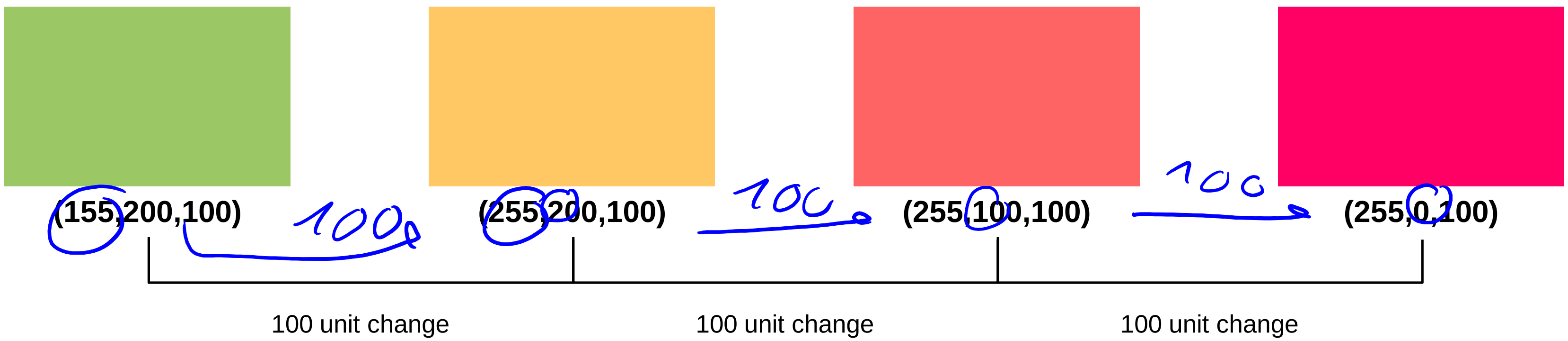


Given the emitted or reflected spectrum of light of an observed point $f(\lambda)$, we perceive 3 (4) values for each cone type (and rod). To compute the intensity, we apply the sensitivity filter of the cones (e.g., $c_{red}(\lambda)$) to the observed spectrum (multiplication) and integrate the result over all wavelengths. For instance, for red this is:

$$red = \int_0^\infty f(\lambda) \cdot c_{red}(\lambda) d\lambda$$

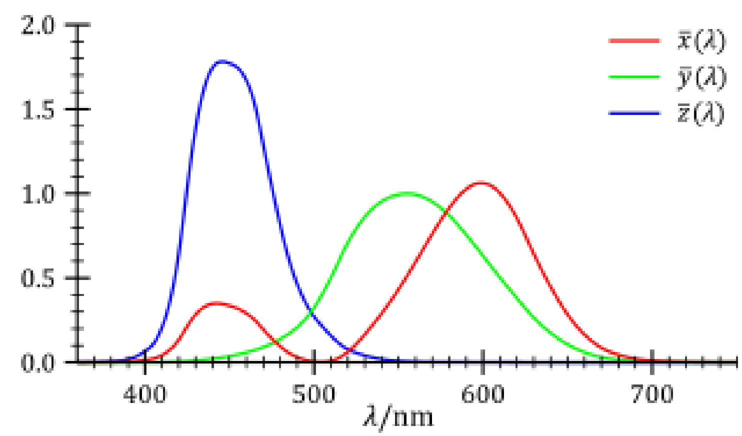
- On the other side, this approximation allows us to artificially re-create the perception with using only 3 additive components emitting wavelengths that match the sensitivity of the red, green, and blue cones. These 3 components form the basis of the RGB family which is optimized for human perception but may not work for the eyes of animals (different sensitivity ranges; for birds with tetrachromatic perception, the UV range is missing).

- Before we can extract features, we need to find a good representation for color that matches human perception. Consider the four colors below in the sRGB space. Between two neighboring boxes, the color distance is 100 units (only one channel changes). Even though the distance is the same, we perceive the color changes differently. The change from green to yellow (1st and 2nd) is significant, while the change from red to pink (3rd to 4th) is smaller. The reason is the non-linear interpretation of sRGB space as we process the light emission from the monitor (or from the reflection of the paper).



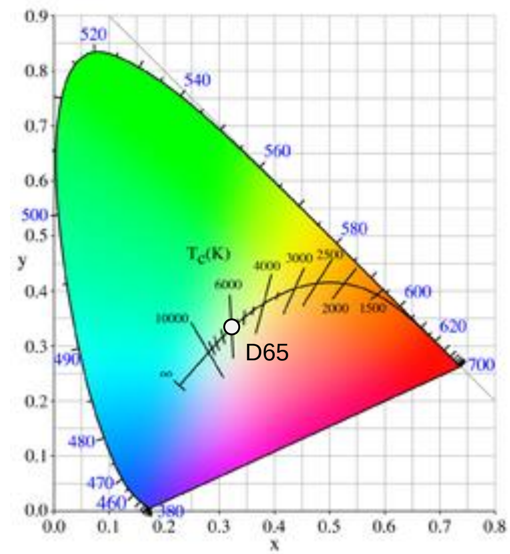
- There are five major color systems (we only look at the first three models subsequently)
  - **CIE** – created by the International Commission on Illumination (CIE) to define a relation between the physical signal and the perception of a (standard) human observer
  - **RGB** – the dominant system since the definition of sRGB by HP and Microsoft in 1996
  - **HSL/HSV** – which translates the cartesian RGB coordinates to cylindrical coordinates for hue and saturation, and uses luminance/brightness as third component
  - **YUV** – used in NTSC and PAL signals and basis of many image and compression algorithms such as JPEG and MPEG (using YCbCr) *[not discussed subsequently]*
  - **CMYK** – used in printing to subtract color from an initially white canvas. The ink absorbs light and a combination of different inks produces the desired color *[not discussed subsequently]*

- The CIE defined a series of color spaces to better describe perceived colors of human vision. The mathematical relationships are essential for advanced color management.
  - The **CIE XYZ** space was defined in 1931 as an attempt to describe human perceived colors. In their experiments, they noted that observers perceive green as brighter than red and blue colors with the same intensity (physical power). In addition, in low-brightness situations (e.g., at night) the rods dominate with a monochromatic view but at much finer resolution of brightness changes.
    - The definition of X, Y and Z does not follow the typical approach of additive or subtractive primary colors. Instead, Y describes the luminance while X and Z describe chromaticity regardless of brightness. Y follows the sensitivity for the M-cones (green), Z the one of the S-cones (blue), and X is a mix of cone responses.
    - To compute X, Y, and Z from spectral data, a standard (colorimetric) observer was defined based on extensive experiments. This represents an average human's chromatic response within a 2 degree arc inside the fovea (central vision; cones mostly reside inside this area). The color matching functions $\bar{x}(\lambda), \bar{y}(\lambda)$ and $\bar{z}(\lambda)$ describe the spectral weighting for the observed spectral radiance or reflection $f(\lambda)$. We obtain the values for X, Y, and Z as follows (note that the spectrum is reduced to the range 380nm to 780nm):



$$X = \int_{380}^{780} f(\lambda) \cdot \bar{x}(\lambda) d\lambda \qquad Y = \int_{380}^{780} f(\lambda) \cdot \bar{y}(\lambda) d\lambda \qquad Z = \int_{380}^{780} f(\lambda) \cdot \bar{z}(\lambda) d\lambda$$

- The three cone types of human vision require 3 components to describe the full color gamut. The concept of color can be divided into different aspects:
  - Brightness – visual perception of the radiating or reflected light and dependent on the luminance of the observed object. It is, however, not proportional to the luminance itself, instead it is an interpretation subjective to the observer.
  - Chromaticity – objective specification of the color in absence of luminance. It consists of two independent components, hue and saturation. Chromaticity diagrams depict the visible or reproducible range of colors. The standard chart is depicted on the right side.
  - Hue – describes the degree a color matches the perception of red, green, blue, and yellow. The hue values are on the boundary of the chromaticity diagram and is usually measured as a degree from the neutral white point (e.g., D65). Red corresponds to 0, yellow to 60, green to 120, and blue to 240.



  - Saturation / Chroma / Colorfulness – measure how much the light is distributed across the visual spectrum. Pure or saturated colors focus around a single wavelength at high intensity. To desaturate a color in a subtractive system (watercolor), one can add white, black, gray, or the hue's complement. In the chromaticity diagram, saturation is the relative distance to the white point. Relative means in terms of the maximum distance in that direction. Note that green is much farther away from white than red and blue.
- The CIE then defined a series of color models to better capture the above components of color perception. We consider in the following the CIE xyY, Lab, and LCH model.

– The CIE xyY space, defined in 1931, was the first attempt to isolate chromaticity from luminance. The Y value of CIE XYZ was created in such a way that it represents perceived luminance of the standard observer. The x, y and z components are derived through a normalization

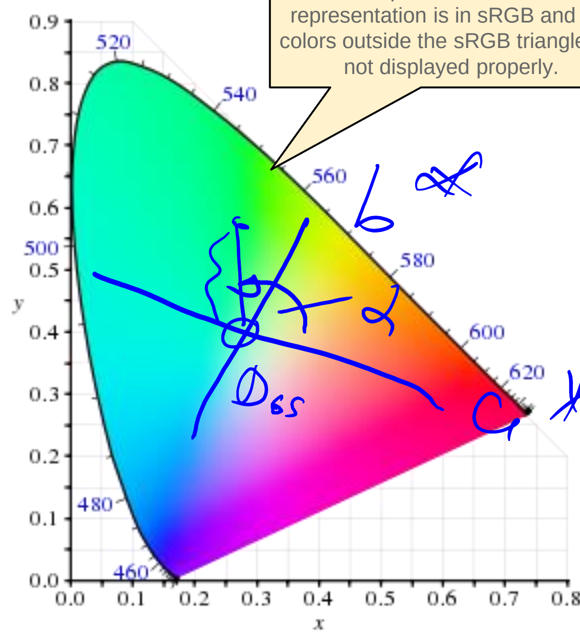
$$x = \frac{X}{X+Y+Z} \qquad y = \frac{Y}{X+Y+Z} \qquad z = \frac{Z}{X+Y+Z} = 1 - x - y$$

The derived color space consists of x, y, and Y. The x, y values define the chromaticity diagram as shown in the lower right part of the page (color in absence of luminance). CIE xyY is widely used to specify color. It encompasses all visible colors of the standard observer. Note that the pictures of the chromaticity diagram here is depicted in the sRGB space an hence does not show the full gamut of the space. Given the x, y and Y values, the back transformation is as follows:

$$X = \frac{Y}{y}x \qquad Z = \frac{Y}{y}(1 - x - y)$$

The outer curve of the chromaticity diagram, the so called spectral locus, show wavelengths in nanometer. The CIE xyY space describes color as perceived by the standard observer. It is not a description of the color of an object as the perceived color of the object depends on the lightning and can change depending on the color temperature of the light source. In dim lightning, the human eye looses the chromaticity aspect and is reduced to a monochromatic perception.

Chromaticity diagram of CIE xyY color space. Note that this representation is in sRGB and the colors outside the sRGB triangle are not displayed properly.



– CIE xyY spans the entire color gamut that is visible for a human eye, but it is **not perceptually uniform**: the perceived difference between two colors with a given distance apart greatly depends on the location in the color space. The **CIE L*a*b*** color space is a mathematical approach to define a perceptually uniform color space. It exceeds the gamut of other color spaces and is device independent. Hence, it is frequently used to map color from one space to another.

- The L component denotes lightness. It depends on the luminance Y but adjusted to perception to create a uniform scale (1 unit difference is perceived as the same lightness change). It typically ranges between 0 and 100, with $L = 0$ representing black, and $L = 100$ being white.
- The a^* component represents the red/green opponents. Negative values correspond to green, while positive values correspond to red. The values often range from -128 to 127. $a^* = 0$ denotes a neutral gray.
- The b^* component represents the blue/yellow opponents. Negative values correspond to blue, while positive values correspond to yellow. The values often range from -128 to 127. $b^* = 0$ denotes a neutral gray.

The transformation from X, Y, Z components under illuminant D65 and $0 \leq Y \leq 255$ is:

$$L^* = 116 \cdot f\left(\frac{Y}{Y_n}\right) - 16$$

$$a^* = 500 \cdot \left(f\left(\frac{X}{X_n}\right) - f\left(\frac{Y}{Y_n}\right)\right)$$

$$b^* = 200 \cdot \left(f\left(\frac{X}{X_n}\right) - f\left(\frac{Z}{Z_n}\right)\right)$$

$$f(t) = \begin{cases} \sqrt[3]{t} & \text{if } t > \left(\frac{6}{29}\right)^3 \\ \frac{841 \cdot t}{108} + \frac{4}{29} & \text{otherwise} \end{cases}$$

$$X_n = 242.364495 \qquad Z_n = 277.67358$$

$$Y_n = 255.0$$

- The **CIE LCH** differs from CIE L*a*b* by the use of cylindrical coordinates. $L = L^*$ remains, but a^* and b^* are replaced by the chroma C (saturation, colorfulness) and hue H. Based on the definition of the a^*- and b^*-axis, the center is at the defined white point (e.g., D65). The hue H is then the angle from the a^*-axis (counterclockwise). The chroma C is the distance from the center.

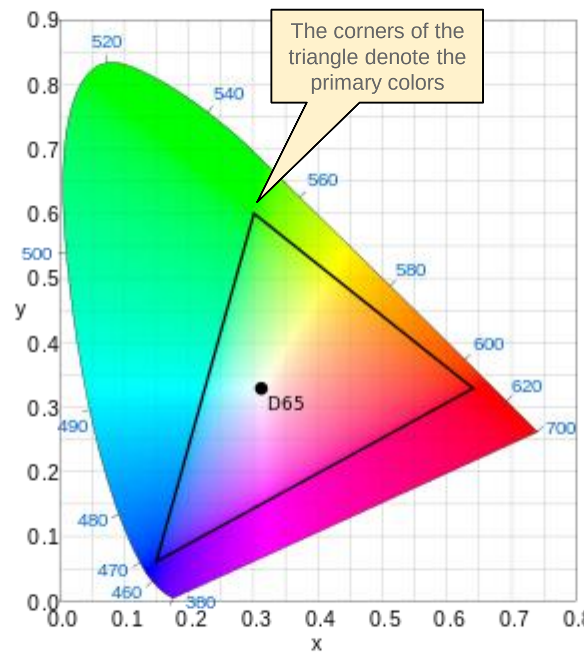
$$L = L^* \qquad C = \sqrt{(a^*)^2 + (b^*)^2} \qquad H = \arctan(a^*, b^*)$$

$\arctan(a^*, b^*)$ is the arc tangent of b^*/a^* taking the quadrant of (a^*, b^*) into account

  - This is not the same as the better known HSL/HSV color models (also use cylindrical coordinates). These models are a polar coordinate transformation of the RGB color space, while CIE LCH is a polar coordinate transformation of CIE L*a*b*.
  - CIE LCH is still perceptually uniform. However, H is a discontinuous function as the angle abruptly changes from 2π to 0. This can cause some issues if the angles are not correctly "subtracted" from each other.
- The CIE has defined further models like the CIE L*u*v*, CIE RGB, and the CIE UVW which we omit here.

- The **RGB** color space is the standard model in computing since HP and Microsoft cooperatively defined sRGB as an additive color model for monitors, printers and the Internet. It has been standardized as IEC 61966-2-1:1999 and is the “default” color model (if the model is not defined).
  – sRGB uses the ITU-R BT.709 (or Rec. 709) primaries to define the color gamut (space of possible colors). The advantage, and mostly the reason for its success, was the direct transfer to a typical CRT monitor at that time. The primaries are:

Chromaticity	Red	Green	Blue	White Point (D65)
x	0.6400	0.3000	0.1500	0.3127
y	0.3300	0.6000	0.0600	0.3290
Y	0.2126	0.7152	0.0722	1.0000



  – For non-negative values, sRGB colors are bound to the triangle depicted in the right-hand figure. Note that the color gamut is not covering all chromaticities, especially a large fraction of the green/blue range is missing.
  – The sRGB scales are non-linear (approximately a gamma of 2.2). To convert from linear RGB to sRGB, the specification provides functions to map channel values. Let c_{sRGB} denote a channel value (red, green, blue) in the sRGB space, and c_{linear} denote a value in linear RGB. Both with ranges between 0 and 1 (for quantized value, divide/multiply by $2^{\text{bits}} - 1$)

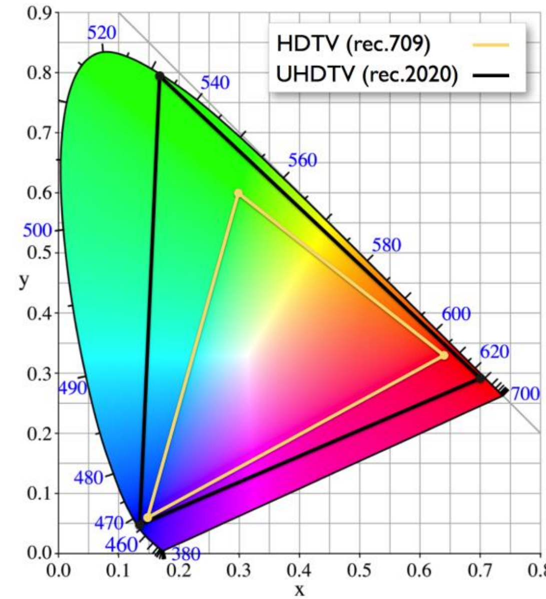
$$c_{sRGB} = \begin{cases} 12.92 \cdot c_{linear} & \text{if } c_{linear} \leq 0.0031308 \\ 1.055 \cdot c_{linear}^{\frac{1}{2.4}} - 0.05 & \text{otherwise} \end{cases} \qquad c_{linear} = \begin{cases} \dfrac{c_{sRGB}}{12.92} & \text{if } c_{sRGB} \leq 0.04045 \\ \left(\dfrac{c_{sRGB} + 0.055}{1.055}\right)^{2.4} & \text{otherwise} \end{cases}$$

- The conversion from CIE XYZ to linear RGB is as follows:

$$\begin{bmatrix} r_{linear} \\ g_{linear} \\ b_{linear} \end{bmatrix} = \begin{bmatrix} 3.240479 & -1.537150 & -0.498535 \\ -0.969256 & 1.875992 & 0.041556 \\ 0.055648 & -0.204043 & 1.057311 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \qquad \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} 0.412453 & 0.357580 & 0.180423 \\ 0.212671 & 0.715160 & 0.072169 \\ 0.019334 & 0.119193 & 0.950227 \end{bmatrix} \begin{bmatrix} r_{linear} \\ g_{linear} \\ b_{linear} \end{bmatrix}$$

  - Note that the transformation above is a mapping between linear RGB and XYZ. To obtain sRGB values, a further transformation is needed (see previous page).
  - Also note that the RGB space is not covering the entire XYZ space and the visible colors of human perception. If the mapping leads to values outside of [0,1], the value is mapped to the closest limit (0 for negative values, and 1 for values ≥ 1).

- RGB values are often quantized to integer ranges. The mapping is simply a multiplication and division by $2^{\text{bits}} - 1$. For true color (32-bit), the multiplier is 255, for deep color (64-bit), the multiplier is 65536. In some cases, quantization is based on 2^{bits} reference colors (color palette). A color is then represented by its nearest neighbor in the palette.
- Next to the sRGB and linear RGB model, various alternatives were defined. In essence, it is simple to construct an RGB space by defining the primaries and the white point. Alternative RGB model extend the original, rather constrained sRGB to a wider range of color gamut. For instance, Rec. 2020 for ultra-high-definition television (UHDTV). It has a much broader color gamut than HDTV which is based on Rec. 709. Some RGB models even excess the chromaticity chart to cover more of the green/blue area.



UHDTV vs HDTV Color Gamuts

- Artists often start with a relatively bright color and than add a) white to “tint” the color, or b) black to “shade” the color, or c) white and black (gray) to tone the color. To enable such techniques in computer graphics, **HSL** and **HSV** color models are alternative representations of the RGB space designed to simplify color making. Both use hue (H) and chroma (S) to define chromaticity. The HSL uses lightness (L) and places fully saturated colors at $L = 1/2$. It allows both tinting ($L \rightarrow 1$) and shading ($L \rightarrow 0$) without change of saturation. HSV uses value (V) and places fully saturated colors at $V = 1$. It allows shading ($V \rightarrow 0$) without changing saturation, but tinting adjusts saturation.

$$H' = \begin{cases} 0 & \text{if } C = 0 \\ \dfrac{G-B}{C} \bmod 6 & \text{if } M = R \\ \dfrac{B-R}{C} + 2 & \text{if } M = G \\ \dfrac{R-G}{C} + 4 & \text{if } M = B \end{cases}$$

$$M = \max(R, G, B)$$

$$m = \min(R, G, B)$$

$$C = M - m$$

$$H = 60° \cdot H'$$

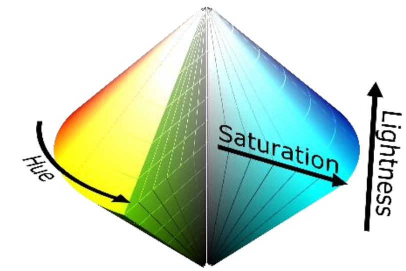
$$V = M$$

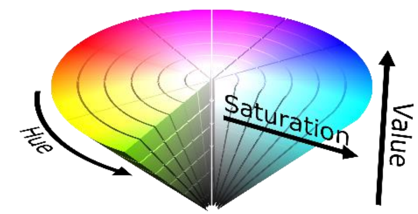
$$S_{HSV} = \begin{cases} 0 & \text{if } V = 0 \\ \dfrac{C}{V} & \text{otherwise} \end{cases}$$

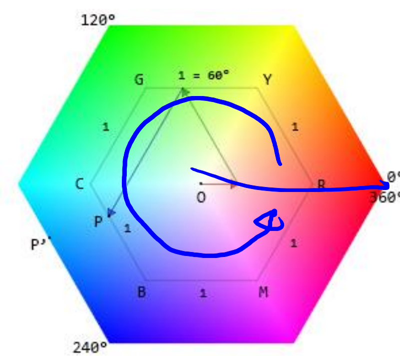
$$H = 60° \cdot H'$$

$$L = \frac{1}{2}(M + m)$$

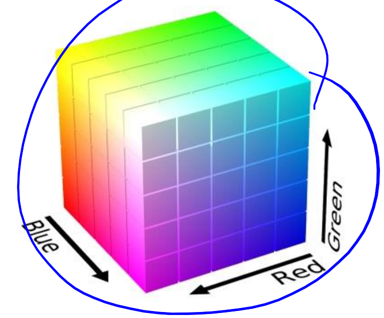
$$S_{HSL} = \begin{cases} 0 & \text{if } L = 1 \\ \dfrac{C}{1 - |2L - 1|} & \text{otherwise} \end{cases}$$



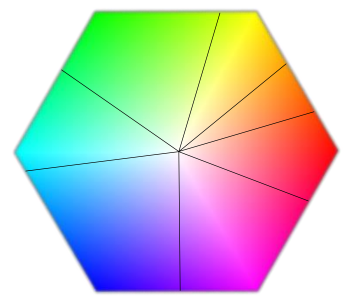




- **Color Histogram**: histograms are a simple way to describe the distribution of colors using a set of reference colors. The fixed reference colors are the "vocabulary" of the collection. The color of each pixel is mapped to the nearest reference color, then we count how often the reference colors occur in the image. To make the feature scale invariant, the counts are normalized by the total number of pixels. The result can also be interpreted as the probability that a reference color occurs. 125
  - Selection of reference colors
    - The most simple way is to quantize the R, G, B values in the linear RGB space as on the right hand side. With 2 bits, for example, we obtain 4 uniform ranges along each channel, and a total of 64 reference colors c_i with $1 \leq i \leq 64$. We can use any number of uniform ranges (e.g., 5) to obtain the desired number of colors.



    - To improve perceptual matching of color, it is better to use a non-uniform distribution. For instance, in the HSV color space, we can divide the color hexagon into areas of perceived similar colors like on the right side. The V-dimension may have more bins to account for the increased brightness sensitivities. With 7 chromaticity values and 9 bins along the V-dimension, we obtain 63 reference colors c_i.



    - If the color space itself is uniform, like in L*a*b*, then we can use uniform ranges. The L^*-axis should have more ranges than the a^*- and b^*-axis to account for brightness sensitivity.
    - We can measure the distance $d_{i,j}$ between reference color c_i and c_j to denote similarities between colors. In cartesian coordinates, this is the Euclidean distance between the centers of the areas representing the colors. In cylindrical coordinates, like the HSV example above, we obtain angle differences as $\min(|\alpha - \beta|, 2\pi - |\alpha - \beta|)$ and apply a Manhattan distance. In all cases, value ranges have to be normalized before distance calculations (e.g., to range [0,1])

- Comparison of histogram (distance measure)
  - Let h_i and g_i denote the normalized histograms of two images ordered by the N reference colors c_i with $0 \le h_i, g_i \le 1$. Note that even though we use a 3-dimensional color space for quantization, the histograms are one-dimensional (through enumeration of reference colors). We also have the distances $d_{i,j} = d_{j,i}$ between two reference colors c_i and c_j.

blue yellow

30%

red orange amber

red orange amber

  - A first naïve approach is to compute a Manhattan (or Euclidean) distance between histograms

$$\delta_{Manhattan}(\boldsymbol{h}, \boldsymbol{g}) = \sum_{i=1}^{N} |h_i - g_i|$$

preferred

$$\delta_{Euclidean}(\boldsymbol{h}, \boldsymbol{g}) = \sqrt{\sum_{i=1}^{N} (h_i - g_i)^2}$$

amber

This distance formulae work quite well, however, they do not take similarity between reference colors into account. A small shift in lightning or color representation can yield large distances.

orange

  - To account for cross-correlation between reference colors, we need to use a quadratic distance measure and use a matrix $\mathbf{A}$ which is based on the distance between reference colors:

$$\delta_{quadratic}(\boldsymbol{h}, \boldsymbol{g}) = (\boldsymbol{h} - \boldsymbol{g})^{\top} \mathbf{A} (\boldsymbol{h} - \boldsymbol{g})$$

$$\mathbf{A}:\; a_{i,j} = 1 - \frac{d_{i,j}}{\max_{k,l} d_{k,l}}$$

Distance normalized by maximum distance for all pairs of reference colors

  - If the user provides a sketch as the query, or the user selects a number of colors that should be present in the picture, histogram intersections (equals to a partial match query) are better suited. Let $g_i \neq 0$ denote the user selected colors and $g_i = 0$ the colors without user input.

$$\delta_{intersection}(\boldsymbol{h}, \boldsymbol{g}) = \frac{\sum_{i=1}^{N} \min(h_i, g_i)}{\min(|\boldsymbol{h}|, |\boldsymbol{g}|)}$$

– Variants:

- A simpler variant is the use of luminance or brightness histograms. The chromaticity aspects are not taken into account. As a first step, brightness or luminance is calculated, for instance, with L^* from CIE L*a*b*. The luminance value is quantized using N uniform ranges. The rest is identical to the approaches above (including quadratic distances to account for similarities between brightness/luminance values). The resulting features describes brightness of the image and is often used for shot detection in videos (different lightning denotes shot boundary)
- Equally, we can only quantize the chromaticity aspects and disregard brightness/luminance. Candidate color spaces are CIE L*a*b, CIE LCH, HSL, or HSV. The resulting features describes color distribution and is invariant to lightning (as long as the lightning does not significantly impact the perception of chromaticity).

– Discussion:

- Histograms are very simple and yield already good results. They are robust against translation, rotation, noise, and scale; in some cases, also against lightning differences.
- The lack of spatial relation between colors may lead to unexpected results. A blue lake (bottom of the picture) will match with a blue sky (top of the picture) and a blue car (middle of the picture). It is simple to construct two images with the same histogram but different content.
- The histogram intersection method is useful to guide a retrieval system to the desired color of (main) objects. The user can pick a color and the search is extended with a histogram sub-query using the intersection method.
- Color histograms tend to have a very high-dimensionality. 64 dimensions is often a minimum for good retrieval, but more than 1000 dimensions can result. Search in such spaces is costly and inefficient. Dimensionality reduction may help to deal with both correlation of reference colors and the reduction of dimensions (see principal component analysis, PCA).

- **Color Moments:** statistical moments are another way to describe the distribution of colors in the selected color space. We can select again any of the color spaces discussed before, but again, to calculate distances and similarities, the perceptual uniform spaces are better suited. We often use L*a*b* as the basis color model (over LCH to avoid the more complicated angular differences)
  - Single channel moments compute statistical parameters for one channel only (L^*, a^*, b^*). Let c denote a color channel, N denote the number of rows, and M the number of columns, then the first four moments are given as:

$$\mu_c = \frac{1}{N \cdot M} \sum_{x,y} c(x,y) \qquad v_c = \frac{1}{N \cdot M} \sum_{x,y} \left(c(x,y) - \mu_c\right)^2$$

$$s_c = \frac{1}{N \cdot M} \sum_{x,y} \left(\frac{c(x,y) - \mu_c}{\sqrt{v_c}}\right)^3 \qquad k_c = \frac{1}{N \cdot M} \sum_{x,y} \left(\frac{c(x,y) - \mu_c}{\sqrt{v_c}}\right)^4$$

    Mean μ_c and variance v_c describe the peak position and width of the peak in the distribution. The skewness s_c describes whether peak is wider to the left or to the right. And Kurtosis k_c denotes the presence of outliers (far away from mean). With three channels, we obtain 12 feature values in this way.

  - We can add additional covariance values between pairs of channels. Let c_1 be a first channel, and c_2 be a second channel. With three channels, we obtain 3 additional covariance value from the possible pairs of channels:

$$cov_{c_1,c_2} = \frac{1}{N \cdot M} \sum_{x,y} \left(c_1(x,y) - \mu_{c_1}\right) \cdot \left(c_2(x,y) - \mu_{c_2}\right)$$

- When calculating the moments, it is possible to transform the formulas such that only one pass is necessary to compute all the values (c denotes a color channel):

$$a_{c,n} = \frac{1}{N \cdot M} \sum_{x,y} c(x,y)^n \qquad 1 \le n \le 4$$

$$b_{i,j} = \frac{1}{N \cdot M} \sum_{x,y} \left(c_i(x,y) \cdot c_j(x,y) \right) \qquad 1 \le i,j \le 3$$

$$\mu_c = a_{c,1} \qquad v_c = a_{c,2} - a_{c,1}^2$$

$$s_c = \frac{a_{c,3} - 3a_{c,2} \cdot a_{c,1} + 2a_{c,1}^3}{v_c^{3/2}} \qquad k_c = \frac{a_{c,4} - 4a_{c,3} \cdot a_{c,1} + 6a_{c,2} \cdot a_{c,1}^2 - 3a_{c,1}^4}{v_c^2}$$

$$cov_{c_i,c_j} = b_{i,j} - \mu_{c_i} \cdot \mu_{c_j}$$

Using the CIE L*a*b* color space, we obtain 12 moments and 3 covariances, a total of 15 feature values. We can combine the values into a vector $\boldsymbol{m}$ (in a defined order) and compare to feature vectors $\boldsymbol{m}_i$ and $\boldsymbol{m}_j$ of two images using either Euclidean or Manhattan distance:

$$\delta_{Manhattan}(\boldsymbol{m}_i, \boldsymbol{m}_j) = \sum_{k=1}^{15} \left| \boldsymbol{m}_{i,k} - \boldsymbol{m}_{j,k} \right| \qquad \delta_{Euclidean}(\boldsymbol{m}_i, \boldsymbol{m}_j) = \sqrt{\sum_{k=1}^{15} \left(\boldsymbol{m}_{i,k} - \boldsymbol{m}_{j,k} \right)^2}$$

- Variants: like with histograms, we can construct moments for brightness/luminance only. Co-variance becomes obsolete and we obtain 4 brightness/luminance moments. We can further construct moments only for the chromaticity aspect, disregarding brightness/luminance. In this case we have 8 moments and one covariance values, resulting in a 9 dimensional feature.
- Discussion:
  - The value ranges of moments vary significantly. Before we can apply a distance measure, we need to scale the values into the same range (e.g., [0,1]). Due to the differences in the distance measure, it is sufficient to just scale the values either by $max - min$ of each component, or the standard deviation of the values along this dimension (not to be confused with the variance color moments; the standard deviation is taken from the actual values along each moment). We can obtain this scaling factors from a large enough sample set and use them as constant factors when extracting the features.
  - Color moments, like histograms, are robust against translation, rotation, noise, and scale; in some cases, also against lightning differences. The lack of spatial relation between colors may lead to unexpected results (like with histograms).
  - In contrast to histograms, the color moments are independent from each other and we do not need a cross-correlation matrix for a quadratic distance function. The resulting vectors are also much shorter (15 if all moments are taken) than the histograms (up to 1000 bins possible). The compact representation leads to obvious performance gains but no loss in retrieval quality.

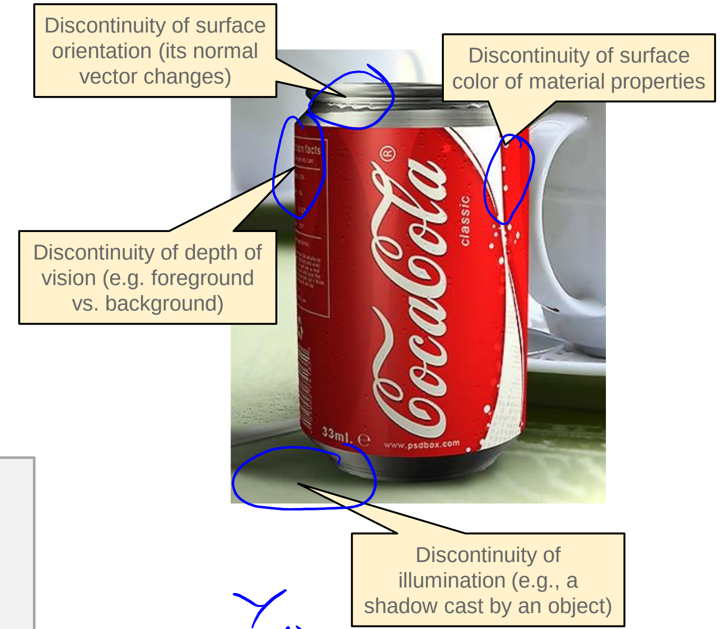
4.4.5 Feature Extraction – Texture Information (Step 3 & 4)

- Texture describe the structure of a surface or part of the image and provides us with information about the spatial arrangement of colors, changes in this arrangement, and the direction and frequency of these changes. We can analyze texture in three ways:
  - Structural approach: Find sets of primitive so-called texels that are composed to regular and repeated patterns as per the examples below:

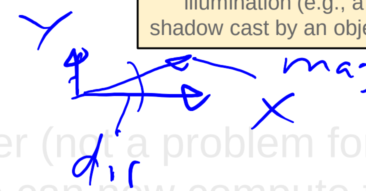
    

    This approach is limited to artificially generated images and does not work for natural images. The inverse problem of creating texture on the surface of objects is well supported by today's graphic processors (see texels, and Voronoi tessellation).
  - Statistical approach: Measure the arrangements in the neighborhood of pixels, quantify them, and create statistical summaries (histograms, moments). We will look at edge detection and optimized filters to get texture features.
  - Fourier approach: Transform the image into the frequency space via Fourier transformation and extract information about the support for so-called Gabor filters in the frequency space.
- Often, we study texture only in grayscale images. For that purpose, we can compute the Y or L^* components in the CIE color models. Recall, that the original picture first needs to be transformed to linear RGB before computing the transformation to CIE XYZ and CIE L*a*b* (see sRGB → linear RGB). In the following, we assume monochromatic images with only a brightness/luminance channel. Advanced methods may also consider chromaticity information for textures.

- **Edge magnitude and direction** (structural approach)
  - Edges in images are caused by several factor as shown on the picture on the right hand side. The detection of edges is the search for gradients with high energy (abrupt change of neighboring pixels). The standard approach is to apply a Sobel operator (convolution) on a smoothed (Gaussian) version of the image, and to determine g_x and g_y values for a pixel. The kernel matrices are given as:



$$\mathbf{G}_x = \frac{1}{8}\begin{bmatrix} +1 & 0 & -1 \\ +2 & 0 & -2 \\ +1 & 0 & -1 \end{bmatrix} \qquad \mathbf{G}_y = \frac{1}{8}\begin{bmatrix} +1 & +2 & +1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$



We can omit the factor 1/8 but then the gradient values are 8 times larger (not a problem for the method shown here). The operators yield a g_x and g_y for each pixel. We can now compute the gradient magnitude $g_{mag}(x,y)$ and the direction of the gradient $g_{dir}(x,y)$ as follows:

$$g_{\mathrm{mag}}(x,y) = \sqrt{g_x(x,y)^2 + g_y(x,y)^2} \qquad g_{dir}(x,y) = \arctan\big(g_x(x,y), g_y(x,y)\big)$$

$\arctan(x,y)$ is the arc tangent of y/x taking the quadrant of (x,y) into account

  - With the above transformation, we obtain 2 values for each pixel in the image. The first value describes how large the change is (energy), the second value represents the direction of change (from darker to lighter). A value of $g_{dir} = 0$ is a vertical edge (change direction is normal to the edge) and the lighter pixel is on the right hand side.

- We now can create simple texture based features.
  - Edgeness of image: Proportion of image with $g_{mag}(x,y) \geq \tau$ for a given threshold τ. This expresses how many edges we can expect on the picture with high enough energy. Continuous areas of them image with, for example, the sky or a lake will result in low values, while several objects or city images with lead to higher values.

$$f_{edgeness} = \frac{1}{N \cdot M} \sum_{x,y} \begin{cases} 1 & \text{if } g_{mag}(x,y) \geq \tau \\ 0 & \text{otherwise} \end{cases}$$

  - Gradient Histograms: same approach as with color histogram. We now have to values per pixels and quantify the direction and the magnitude. The distance between reference gradients is calculated similar as for the HSV color model. Recall that differences in direction are calculated as $\min(|\alpha - \beta|, 2\pi - |\alpha - \beta|)$. We need to normalize energy and direction ranges to compute the distance $d_{i,j}$ between two reference gradients. This allows us to compute the matrix $\mathbf{A}$ for the quadratic distance measure. Given to histograms $\boldsymbol{h}$ and $\boldsymbol{g}$, and assuming N reference gradients, we obtain distances as follows:

$$\delta_{Manhattan}(\boldsymbol{h},\boldsymbol{g}) = \sum_{i=1}^{N} |h_i - g_i|$$

$$\delta_{Euclidean}(\boldsymbol{h},\boldsymbol{g}) = \sqrt{\sum_{i=1}^{N} (h_i - g_i)^2}$$

$$\delta_{quadratic}(\boldsymbol{h},\boldsymbol{g}) = (\boldsymbol{h} - \boldsymbol{g})^\top \mathbf{A} (\boldsymbol{h} - \boldsymbol{g})$$

$$\mathbf{A}\colon\ a_{i,j} = 1 - \frac{d_{i,j}}{\max_{k,l} d_{k,l}}$$

Distance normalized by maximum distance for all pairs of reference colors

As with color histograms, the same issues with high dimensionality occurs.

- Gradient Moments: as before, we compute moments for the magnitude and the direction, and a covariance value for magnitude and direction. Let c denote either magnitude or direction:

$$\mu_c = \frac{1}{N \cdot M} \sum_{x,y} g_c(x,y) \qquad v_c = \frac{1}{N \cdot M} \sum_{x,y} \left(g_c(x,y) - \mu_c\right)^2$$

$$s_c = \frac{1}{N \cdot M} \sum_{x,y} \left(\frac{g_c(x,y) - \mu_c}{\sqrt{v_c}} \right)^3 \qquad k_c = \frac{1}{N \cdot M} \sum_{x,y} \left(\frac{g_c(x,y) - \mu_c}{\sqrt{v_c}} \right)^4$$

$$cov_{mag,dir} = \frac{1}{N \cdot M} \sum_{x,y} \left(g_{mag}(x,y) - \mu_{mag}\right) - \left(g_{dir}(x,y) - \mu_{dir}\right)$$

This results in 9 feature values describing the distribution of gradients.

- **Laws' Texture Energy** (structural approach)
  - Laws texture masks compute 9 values for a pixel in the image to capture various aspects of texture features. The masks are based on 4 prototype vectors:

$$\boldsymbol{v}_{L5} = [1 \quad 4 \quad 6 \quad 4 \quad 1]$$

Level: (Gaussian) center-weighted local average

$$\boldsymbol{v}_{E5} = [-1 \quad -2 \quad 0 \quad 2 \quad 1]$$

Edge: (gradient) responds to step edges

$$\boldsymbol{v}_{S5} = [-1 \quad 0 \quad 2 \quad 0 \quad -1]$$

Spot: (Laplace of Gaussian) detects a spot

$$\boldsymbol{v}_{R5} = [1 \quad -4 \quad 6 \quad -4 \quad 1]$$

Ripple: (Gabor) detects ripples

– From these base vectors, we can compute 16 matrices by multiplication of pairs of prototype vectors. For the instance E5L5, for instance, we obtain the Kernel matrix $\mathbf{G}_{E5L5}$ as follows:

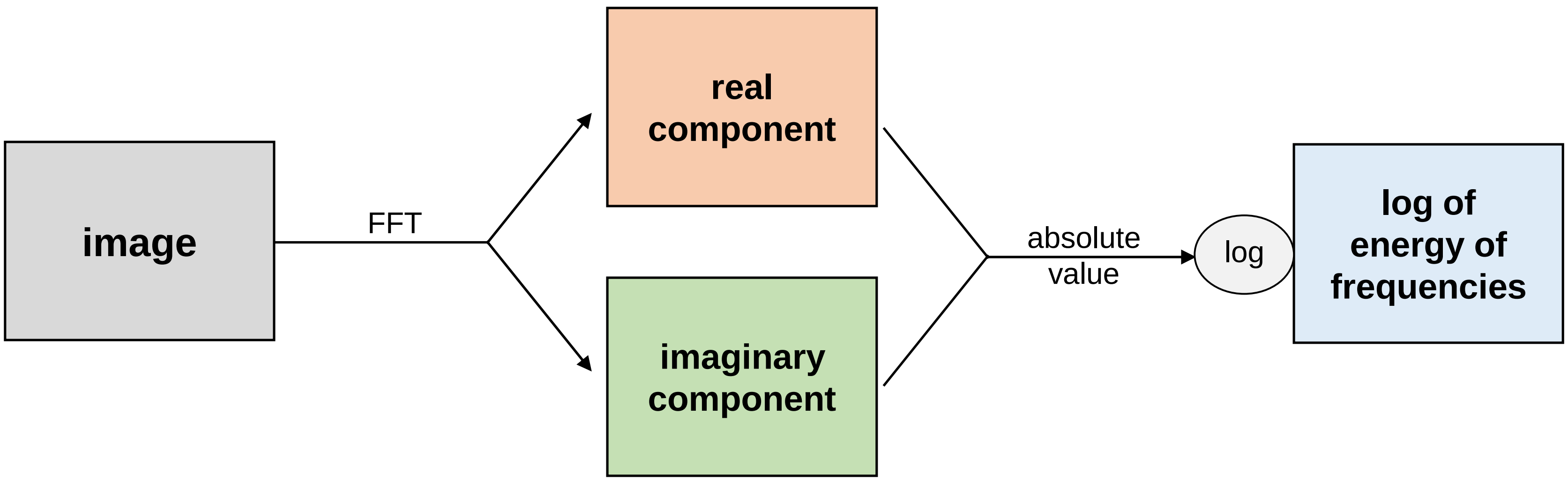
$$\mathbf{G}_{E5L5} = \boldsymbol{v}_{E5}^{\top}\boldsymbol{v}_{L5} = \begin{bmatrix} -1 \\ -2 \\ 0 \\ 2 \\ 1 \end{bmatrix} [1 \quad 4 \quad 6 \quad 4 \quad 1] = \begin{bmatrix} -1 & -4 & -6 & -4 & -1 \\ -2 & -8 & -12 & -8 & -1 \\ 0 & 0 & 0 & 0 & 0 \\ 2 & 8 & 12 & 8 & 2 \\ 1 & 4 & 6 & 4 & 1 \end{bmatrix}$$

Since E5L5 and L5E5 measure a similar aspects, we collapse them into a single Kernel and use the average of both matrices. With such reductions, we obtain 9 Kernel matrices:

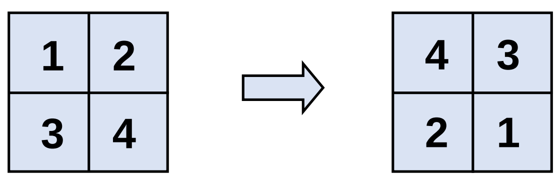
$$\mathbb{G} = \left\{ \frac{\mathbf{G}_{E5L5} + \mathbf{G}_{L5E5}}{2}, \frac{\mathbf{G}_{L5R5} + \mathbf{G}_{R5L5}}{2}, \frac{\mathbf{G}_{E5S5} + \mathbf{G}_{S5E5}}{2}, \frac{\mathbf{G}_{S5L5} + \mathbf{G}_{L5S5}}{2}, \frac{\mathbf{G}_{E5R5} + \mathbf{G}_{R5E5}}{2}, \frac{\mathbf{G}_{S5R5} + \mathbf{G}_{R5S5}}{2} \right\} \cup \left\{ \mathbf{G}_{S5S5}, \mathbf{G}_{R5R5,} \mathbf{G}_{E5E5} \right\}$$

– With these 9 Kernel matrices, we apply a convolution to obtain 9 texture energy values $e_i(x, y)$ per pixel (with $1 \le i \le 9$). From here, we can apply the same approaches as before:
  - Histograms: although feasible, we are faced here with 9 values per pixel. If we quantize them with 4 ranges, we obtain $4^9 = 262{,}144$ reference energies. This clearly exceeds our expectations of a computationally meaningful feature, especially, if we consider the necessity of a quadratic function. Using only 2 ranges yields $2^9 = 512$ reference energies. Acceptable, but the quantification error is significant.
  - Moments: for each energy value, we can calculate 4 moments, and co-variance values for the 36 possible pairs. This yields a 72 dimensional feature vector. If the dimensionality is too high, we can reduce the number of moments (only first 2 or 3) or omit the co-variances.

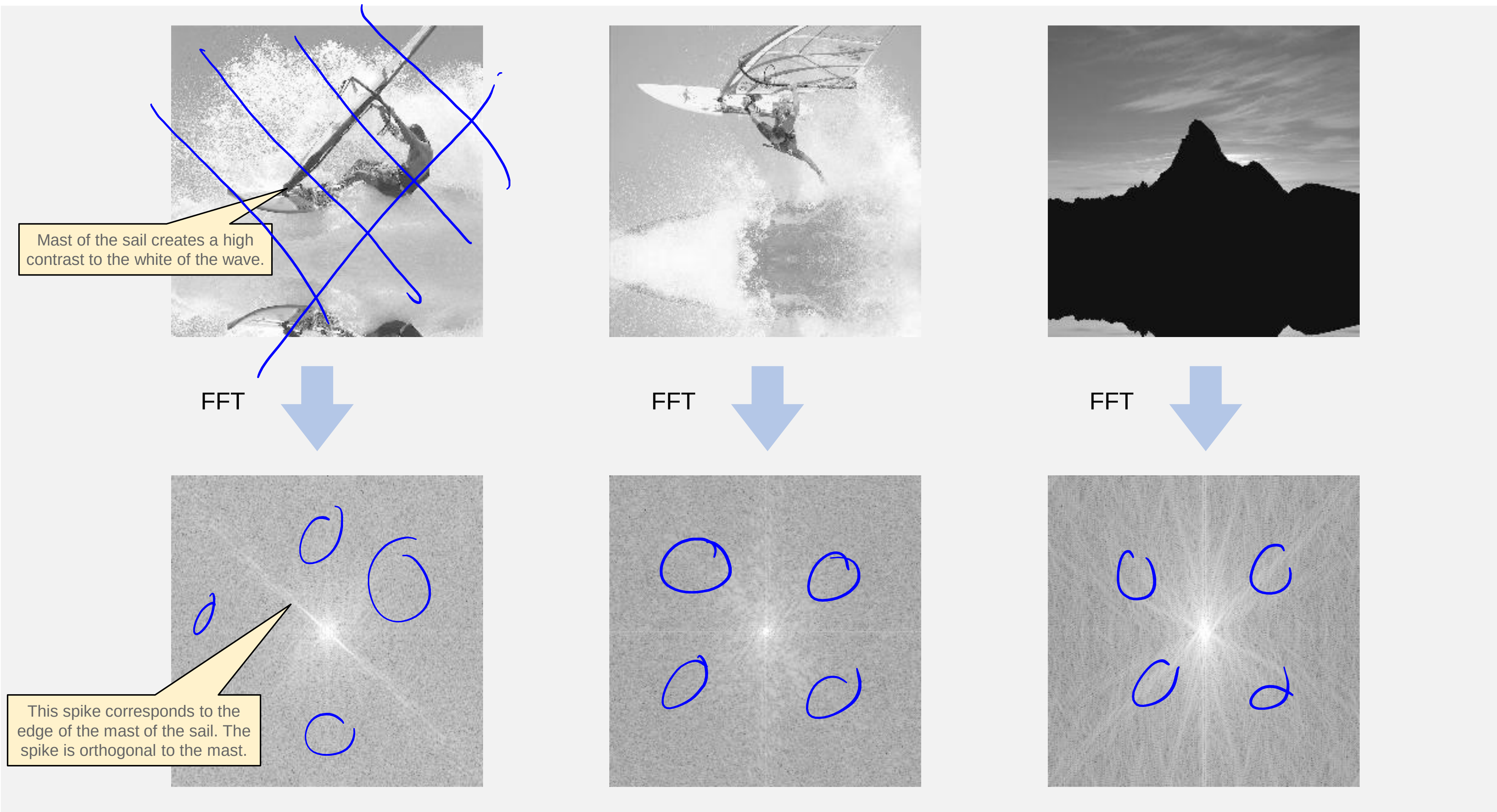
- **Gabor Moments** (Fourier approach)
  - The 2D Fourier transformation maps a (grayscale) image into its frequency space. More formally, it creates a real and imaginary matrix. For the visualizations, we can compute the log of the sum of squared components (the log-function helps for visualization of the large differences in energy). The 2D Fast Fourier Transformation is an accelerated version of the algorithm reducing computational efforts significantly. However, it is only applicable to image sizes of $2^a \times 2^b$. The picture bellow depicts the transformation:



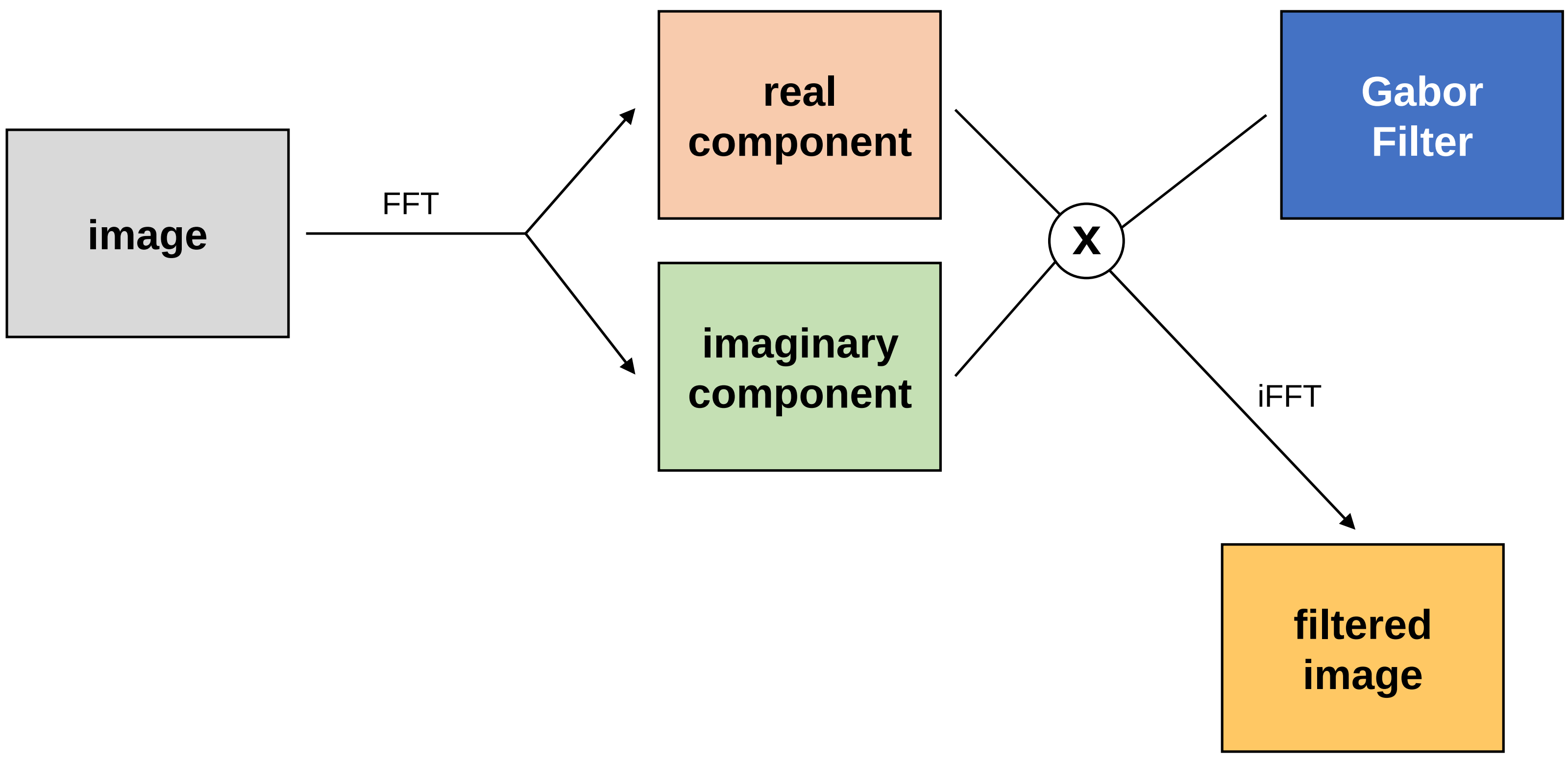
  - To display the frequencies such that low frequencies are in the middle and high frequencies in the outer areas, we need to map the quadrants of the matrix as per below:



– **Examples for the frequency map:** The pictures below show the grayscale original images, and the log-scaled frequency map; the brighter a pixel, the more energy for the corresponding frequency. Low frequencies are in the center, high frequencies in the out areas. The direction from the center to the frequency denotes the normal of an edge in the image for that frequency.



– In the Fourier space, we apply a bank of so-called Gabor filters that select different ranges of frequencies and directions. The Gabor filter is multiplied with the Fourier transformation of the image (a complex matrix), and the result is mapped back via inverse Fourier transformation (here the fast implementation iFFT) to the image space. The filtered image now provides information about the support for the selected frequencies and directions in the original image space. Using banks with 5 orientations and 3 scales, we have 15 Gabor filters and obtain 15 different filtered images. We extract statistical moments for each of these filters to obtain a wide range of texture descriptors. The following pages show the filter banks and its application in the Fourier space.



– The Gabor filter is defined as a Gaussian kernel multiplied by a complex sinusoid. In Neurophysiological experiments, it was shown that the Gabor filters, with the right parameters, behave similar to the receptive fields in the primary visual cortex. Its definition is as follows

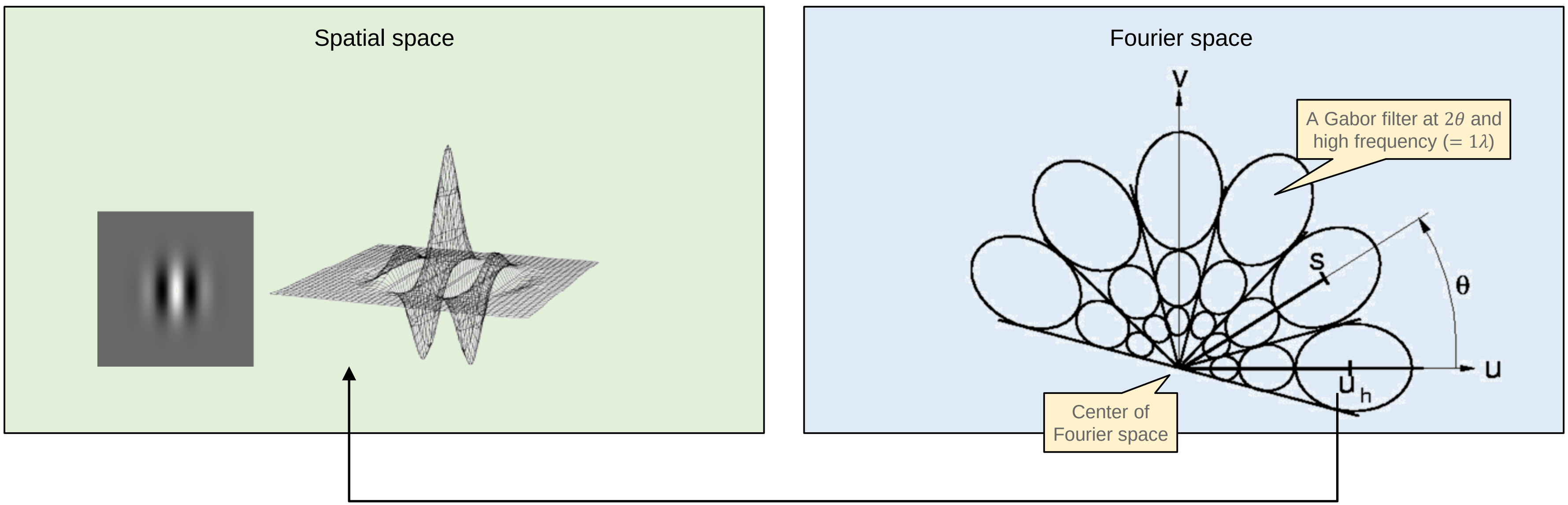
$$g_{\lambda,\theta,\varphi,\sigma,\gamma}(x,y) = e^{-\frac{\bar{x}^2+\gamma^2\bar{y}^2}{2\sigma^2}} \cdot e^{i\left(2\pi\frac{\bar{x}}{\lambda}+\varphi\right)}$$

Gaussian kernel with standard deviation σ and the spatial aspect ration γ

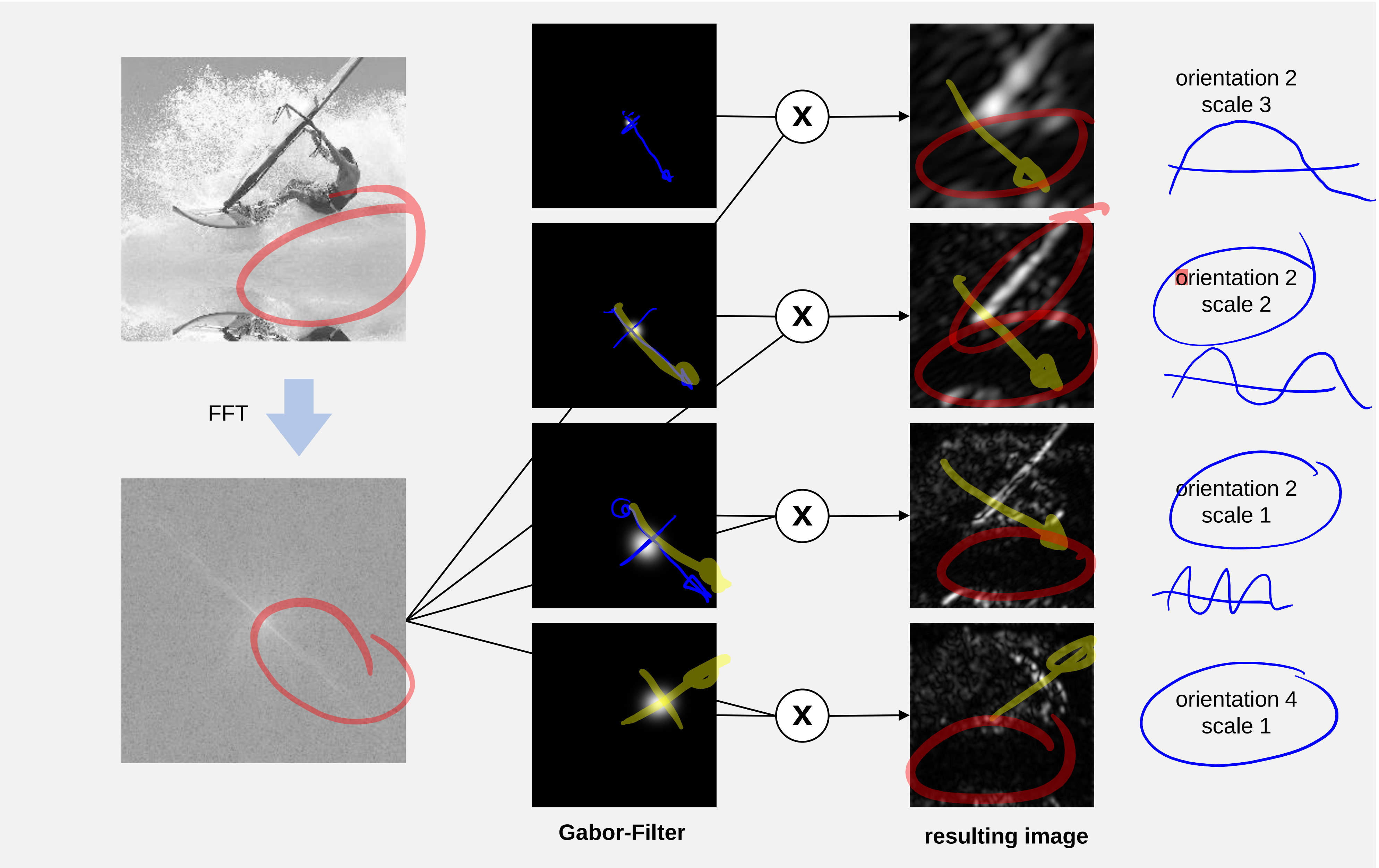
Complex sinusoid with phase φ and wavelength λ. $1/\lambda$ is the frequency of the sinusoid.

$$\bar{x} = x\cos\theta + y\sin\theta$$
$$\bar{y} = -x\sin\theta + y\cos\theta$$

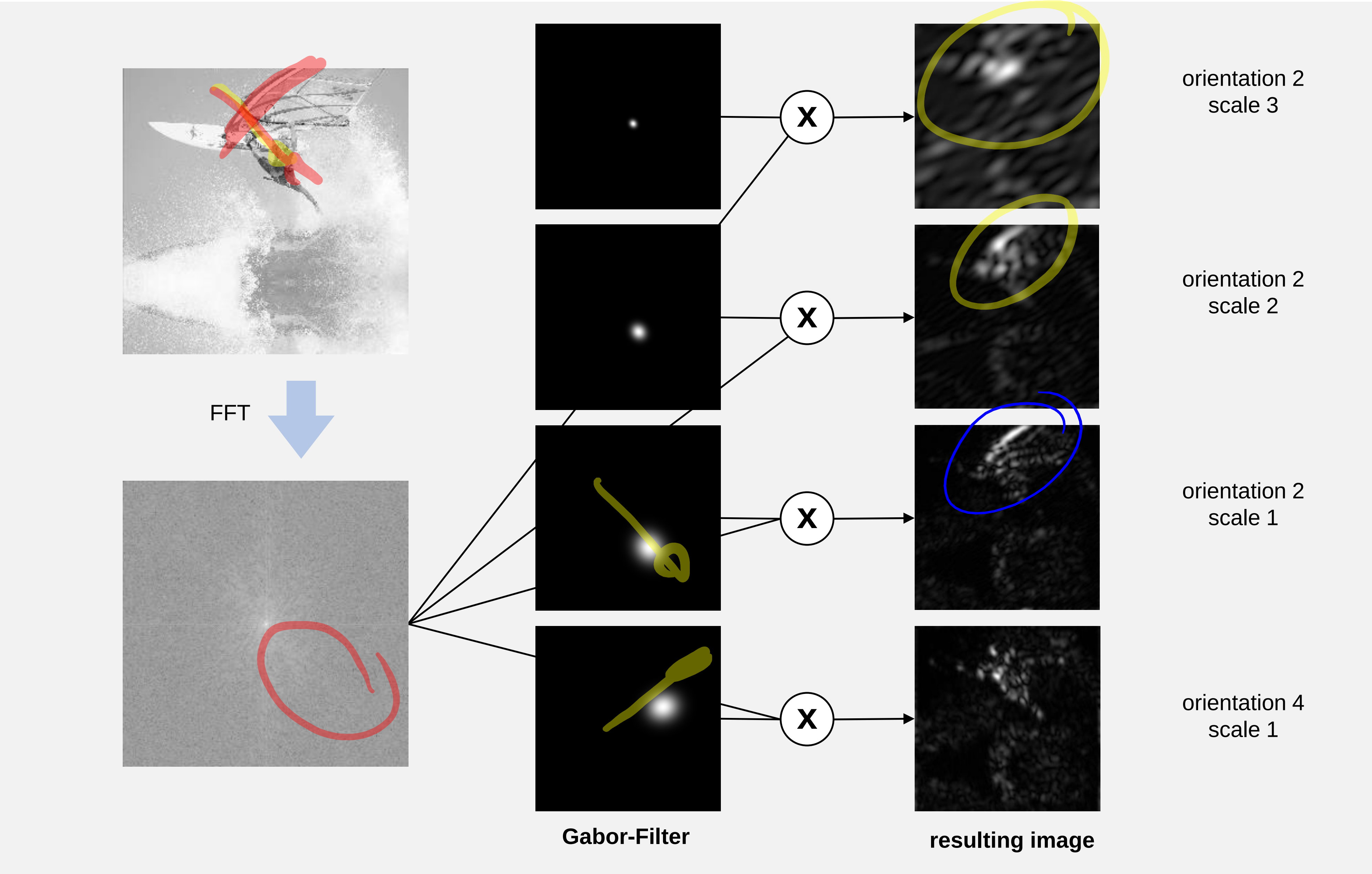
Before application to the Gaussian and sinusoid, the coordinates are rotated by θ. With this definition and varying the parameters, it is possible to construct various filters that are sensitive to frequencies and direction. Mapping the Filter bank into the Fourier space leads to the following layout:



- **Example (1)**

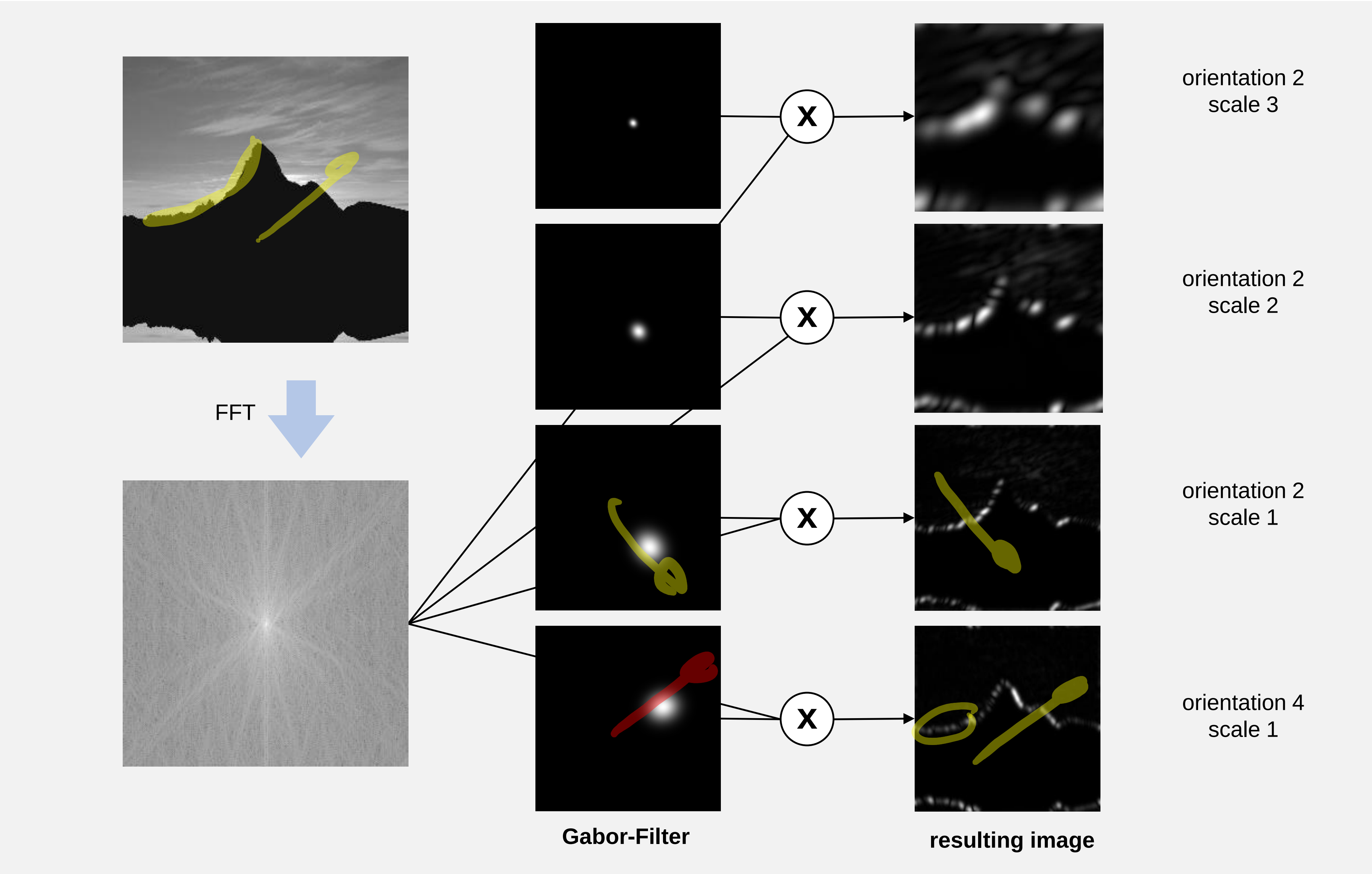

FFT
X
X
X
X
orientation 2
scale 3
orientation 2
scale 2
orientation 2
scale 1
orientation 4
scale 1
Gabor-Filter
resulting image

- **Example (2)**



FFT
X
X
X
X
orientation 2
scale 3
orientation 2
scale 2
orientation 2
scale 1
orientation 4
scale 1
Gabor-Filter
resulting image

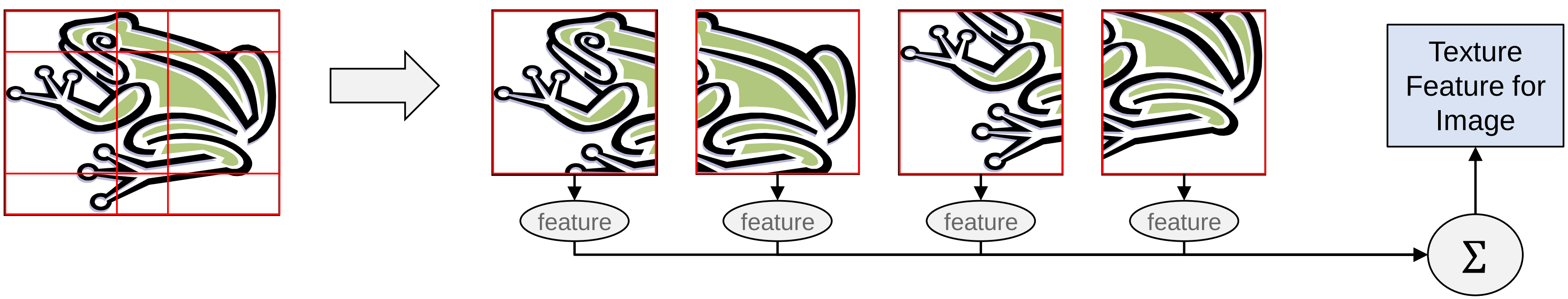
- **Example (3)**


FFT
orientation 2
scale 3
orientation 2
scale 2
orientation 2
scale 1
orientation 4
scale 1
Gabor-Filter
resulting image

– There are two approaches to compute Gabor filtered images:
  - **Fourier space:** compute the Gabor filters in the Fourier space and apply them to the Fourier transformed image. To enable the use of FFT, the size of the image is scaled to the next higher $2^a \times 2^b$ dimension with one of the following methods
    – **Stretching**: stretch the image to match the new size. This changes proportions and thus frequencies and directions in the image.
    – **Filling**: copy the image 1:1 and fill the remaining area with a neutral color.
    – **Tiling**: create a 2-by-2 tile of the same image and crop to the new size.
    – **Mirroring**: create a 2-by-2 tile, but mirror the image at the middle axis. This reduce hard edges that otherwise become visible as spikes. But it adds wrong directions.

Original   Stretching   Filling   Tiling   Mirroring

A further alternative: we use the next smaller $2^a \times 2^b$ dimension and apply the method 4 times for the $2^a \times 2^b$ areas in each corner. At the end, we average all feature values across all areas.



- **Image/spatial space**: compute a Gabor filter bank and apply it to the image through convolution. Since the Gabor filter is complex, we take absolute values of the resulting complex numbers to map back to real numbers. Most image processing libraries (OpenCV, scikit-image) provide implementations for Gabor kernels.

– Once we have the filtered images (like shown in the right hand columns on the pages before with the image examples), we can summarize the results with the usual approaches of histograms or moments. We typically select 3-7 directions ($0 \leq \theta \leq \pi$) and 2-5 scales (or frequencies; $1/\lambda$ usually measured in pixels and ranging from 0.05 to 0.5). With a large number of filters, the moments are again a better choice to reduce the number of dimensions and avoid the complexity of quadratic distance functions.

- With moments, we simply treat the absolute values in the filtered image as the raw data points and compute mean, variance, skewness, and Kurtosis on these values. To further reduce the number of dimensions, it is possible to select only the first 2 or 3 moments. Let $\tilde{f}_i(x,y)$ be the filtered (complex) image representation after applying the i-th Gabor filter. We obtain:

$$\mu_i = \frac{1}{N \cdot M} \sum_{x,y} \left| \tilde{f}_i(x,y) \right| \qquad v_i = \frac{1}{N \cdot M} \sum_{x,y} \left(\left| \tilde{f}_i(x,y) \right| - \mu_i \right)^2$$

$$s_i = \frac{1}{N \cdot M} \sum_{x,y} \left(\frac{\left| \tilde{f}_i(x,y) \right| - \mu_i}{\sqrt{v_i}} \right)^3 \qquad k_i = \frac{1}{N \cdot M} \sum_{x,y} \left(\frac{\left| \tilde{f}_i(x,y) \right| - \mu_i}{\sqrt{v_i}} \right)^4$$

The overall feature is simply the concatenation of all moments across all filters.

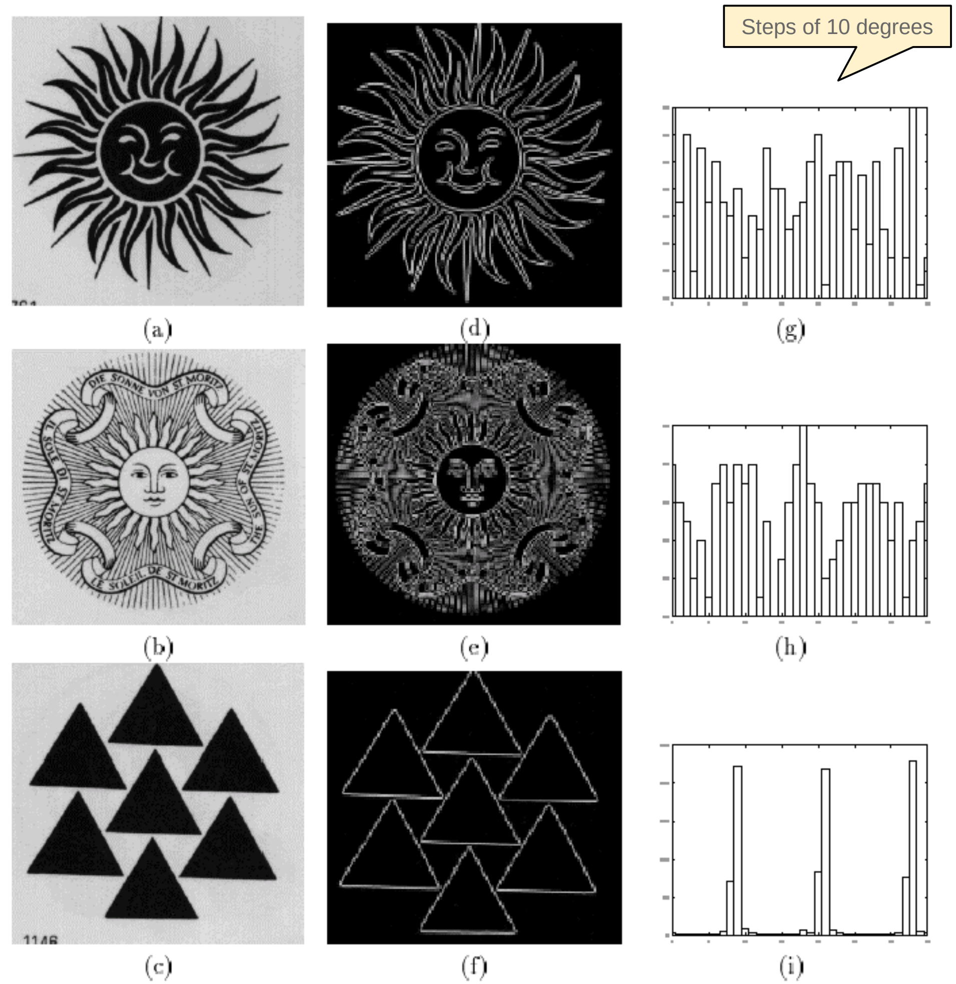
4.4.6 Feature Extraction – Shape Information (Step 3 & 4)

- In this section, we consider three approaches to define shape features.
  - Identify key shape related features in the entire image. There are no segments or objects taken into account, i.e., the features are global for the image.
  - Given a segmentation of the image into objects/blobs, describe the shape of this region to retrieve similar shape from the database. This also works for 2D/3D objects.
  - Identify key points of interest in the picture and describe these points to identify similar objects. This method is used for stitching of panorama images, object recognition, and motion detection.

- **Global Features:** very similar to the texture features, but we are more interested in the contours and direction of these contours than the rest of the image. The basic idea is to apply an edge detector to obtain the outlines of the principle shapes of the image. The Canny edge detector is a solid reference detector with 5 phases (the first two steps are the same as before with texture):

1. Apply Gaussian filter to smooth the image and to remove noise or compression artifacts
2. Compute gradients with their magnitude and direction (as seen before, Sobel operators)
3. Eliminate values that are not a local maximum in the positive/negative direction of the gradient
4. Identify strong edges (magnitude above high threshold) and weak edges (magnitude between low and high threshold) and eliminate values below low threshold.
5. Track edges and eliminate isolated weak edges. Keep only weak edges if in their immediate proximity, there is a strong edge.

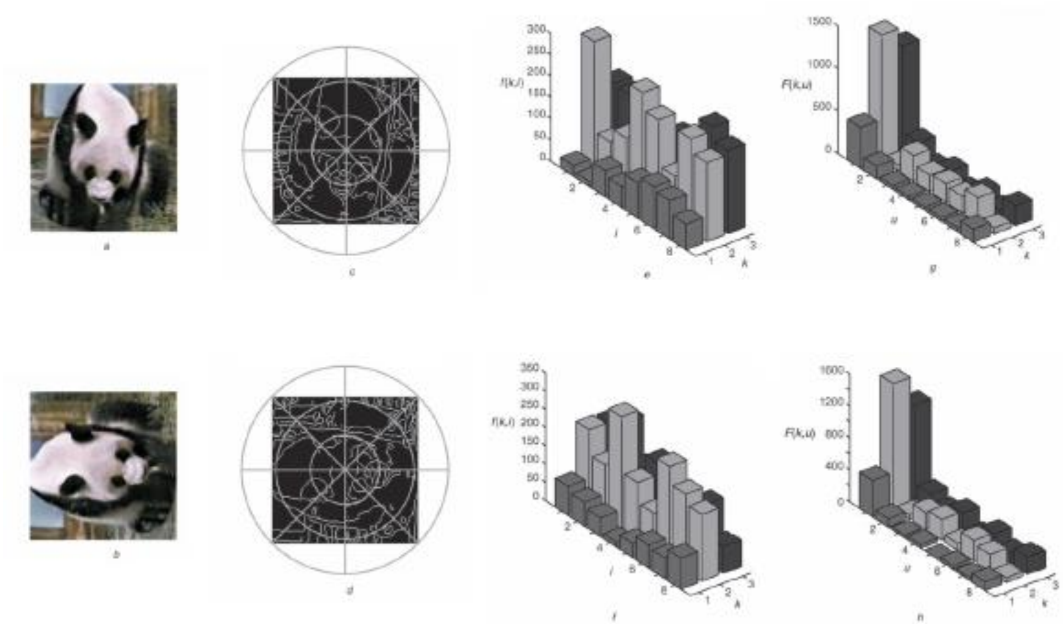
- With the edges, we now can summarize the directions of these edges (the magnitudes have been eliminated in the process) with histograms. The examples on the right side are from an early prototype by Vailaya (1996), Michigan State University.
  - The histograms are normalized by the number of edge pixels and sum up to 1. The step size was 10 degrees hence 36 bins for the histograms.
  - Comparison between histograms is based on the usual distance function. Again, a quadratic distance function is recommended to account for the similarity between angles
  - The feature is translation and scale invariant. With appropriate normalization of the image, we can achieve lightning invariance. However, it is not rotational invariant.
  - To obtain rotational invariance, we need to determine the principle direction and rotate the image such that the principle direction points, for example, upwards. The principle direction is the weighted sum of the original gradients, with the magnitude as weights.



- The **Angular Radial Partitioning** by Chalechale (2003) follows a similar approach to detect edges but uses a different approach to create histograms. The method has 5 steps
  1. Convert the images to grayscale, e.g., by mapping pixels to the L*-channel
  2. Normalize size of images to obtain comparable numbers
  3. Apply Canny edge detector to find strong edges in the image
  4. Partition the resulting edge-map into $M \times N$ radial angular partitions. M is the number of radial sectors, N the number of slices
  5. Count the number of edge pixels in each partition to obtain a raw histogram
  6. Apply a Fourier transform to the histogram and use absolute values (energy) to obtain the final feature vector

  The method is depicted on the right side with an example from the paper.

  - The feature is robust against translation and scale due to initial normalization process. It is robust to small rotational changes as only few pixels will change the partition.
  - The feature is robust against discrete rotations of the angle of the slice due to the Fourier transformation.
  - The feature is robust against omissions of smaller details and noise during edge detection.



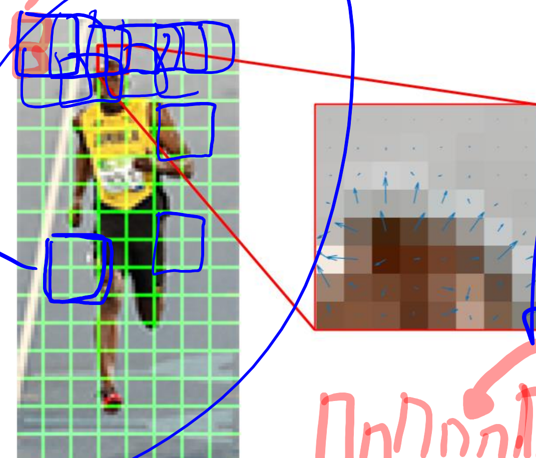
– The **Histogram of oriented gradients** dates back to 1986 but regained interest with the work of Dalal and Triggs in 2005 to detect pedestrians. The methods has since been extended and is often used as input into neural networks.

- Step 1: compute gradients, for instance, with Sobel operators on a grayscale version of the image. In contrast to other approaches, HOG uses unsigned gradients, i.e., the direction lies in the range of 0 to π. Values between π and 2π are rotated by π. Some HOG implementation let users choose between unsigned and signed gradients, but Dalal and Triggs found that this worked best for pedestrian detection
- Step 2: As shown in the picture below, the image is divided into cells each with 8x8 pixels. For each of the cell, HOG computes a 9-bin histogram (9 was found to be optimal for their use case) over the gradient directions of the 64 pixels and weighted by their gradient magnitudes.
- Step 3: gradient magnitudes are variant to illumination and hence require normalization before we can compare histograms with each. Rather than normalizing the 9-bin histograms at each cell, HOG combines 4 neighboring cells and normalizes the concatenated histograms (now 36 bins) so it sums up to 1. The 4 neighboring cells (2x2 cells, each with 8x8 pixels) are moved along the image in steps of 8 pixels. Each block yields a normalized histogram of 36 bins. These blocks are partially overlapping.

$\Sigma = 1$

36 values

8x8



2	3	4	4	3	4	2	2
5	11	17	13	7	9	3	4
11	21	23	27	22	17	4	6
23	99	165	135	85	32	26	2
91	155	133	136	144	152	57	28
98	196	76	38	26	60	170	51
165	60	60	27	77	85	43	136
71	13	34	23	108	27	48	110

Gradient Magnitude

80	36	5	10	0	64	90	73
37	9	9	179	78	27	169	166
87	136	173	39	102	163	152	176
76	13	1	168	159	22	125	143
120	70	14	150	145	144	145	143
58	86	119	98	100	101	133	113
30	65	157	75	78	165	145	124
11	170	91	4	110	17	133	110

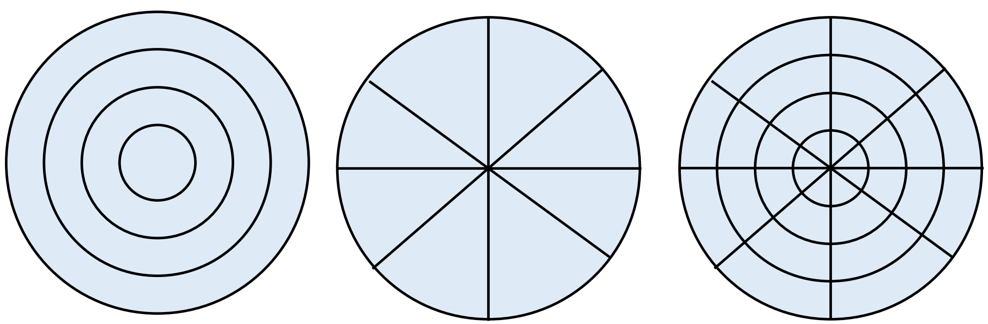
Gradient Direction

9 hist

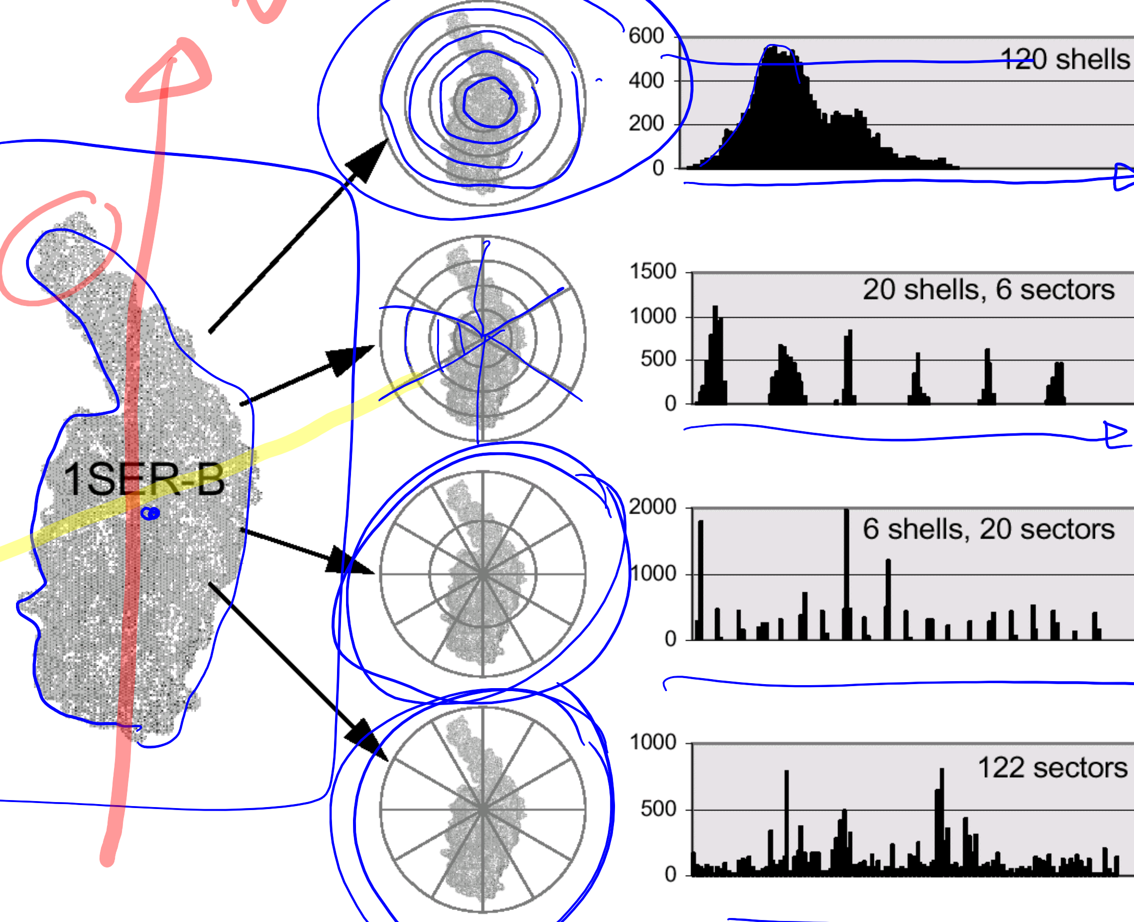
- Step 4: combine histograms to global features or keep a “bag” of local features for search.
- Optional: The HOG features can be used as input into machine learning algorithm. Dalal and Triggs used an SVM to detect pedestrians.

- **Descriptions of blobs/regions/objects**: given a set of segments, blobs or objects, we can describe the regions based on a set of simple spatial metrics. Due to different resolutions and the absence of a standard size of a pixel (unless provided by the image format), spatial metrics are often in relation to the entire image. For example:
  - Area: percentage of pixels within the segment (over the entire image)
  - Centroid: average of all x-values and of all y-values in the region (in absence of mass values)
  - Axis of Least Inertia: this is the axis which allows the rotation of the object with least energy. It is given by the line that minimizes the squared distances to the boundary of the region. This can be used to normalize regions into a primary direction
  - Eccentricity: given a bounding box in the principle direction, the ratio of length to width of the box denotes the eccentricity
  - Circularity Ratio: how closely the shape resembles a circle. There are different definitions, for instance, the ratio of the area of the smallest circle containing the region to the area of the region
  - …and many more

  An alternative approach is to normalize the position of the region (principle direction points upwards) and to measure the overlap with a predefined grid to compute histograms. The histogram values are the relative area covered by the grid. There are different ways to define the grid, for instance:
  - The grid is always such that it contains the region and is a small as possible.
  - With the circular structures, the center is the center of gravity, and the radius is the largest distance of a point to the center of gravity.



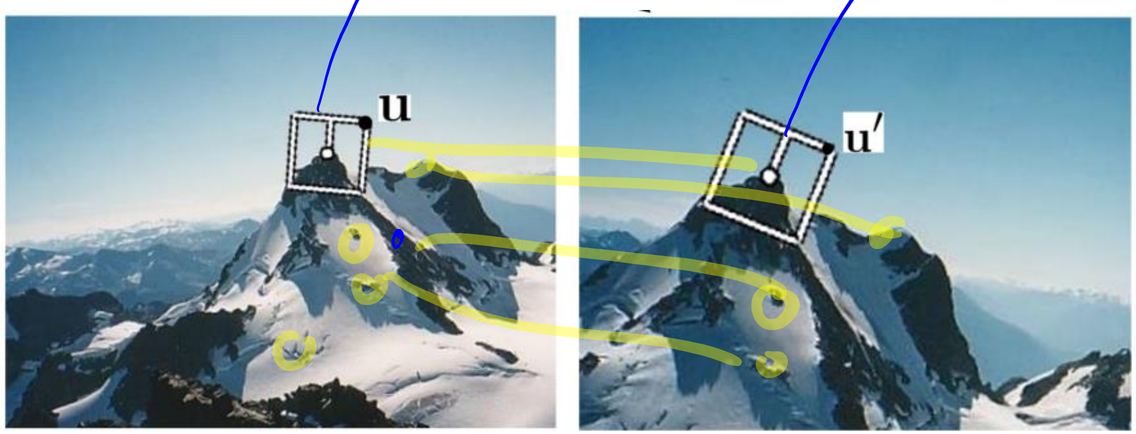
- **Ludwig-Maximilians University Munich** (Berchtold, 1997) studied methods to compare and index 2D and 3D objects. But the methods are similarly applicable to recognized segments in an image. The example on the right side shows 2 complex molecule structure normalized in direction. The partitioning methods extract 4 different histograms, each with 120-122 bins. This is the description of the structure and can be used in combination with a distance measure to find similar objects in the database
  - To make the feature scale invariant, the histogram bins are normalized to sum up to 1.
  - Some of the features are rotation invariant (like the first partitioning). With the initial normalization to a principle direction, rotation invariance is given for all partitioning scheme.
  - The feature is translation invariant due to the use of the center of gravity.



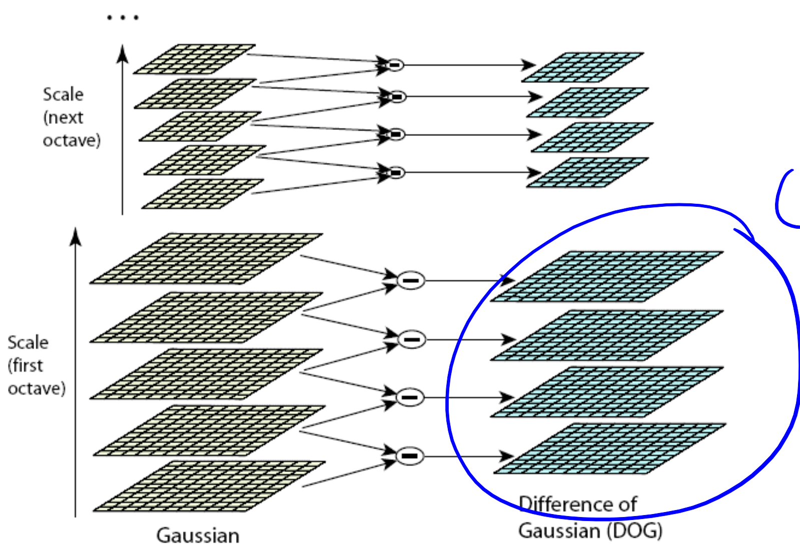
- **Key Points of Interests:** There are many approaches but we consider here only the **Scale Invariant Feature Transform (SIFT).** Due to the complexity of the approach, we summarize the main steps to identify key points of interest and consider how to describe these points to find matches. SIFT extracts features in a very robust way, so that they match again even after significant viewpoint changes. SIFT is used for object recognition, image stitching, motion tracking, and many other use cases, The images below depict the same mountain from slightly different perspective. SIFT is able to match the two highlighted key points despite rotation and scale differences.
  - The algorithms works roughly in 4 steps
    1. Identify scale-space extrema using band-pass filters (difference of Gaussians, DOG)
    2. Keypoint localization with scale; these are the resulting points of interest
    3. Orientation assignment (primary direction of the region around a keypoint for normalization)
    4. Keypoint descriptors that can be used for similarity search

similar

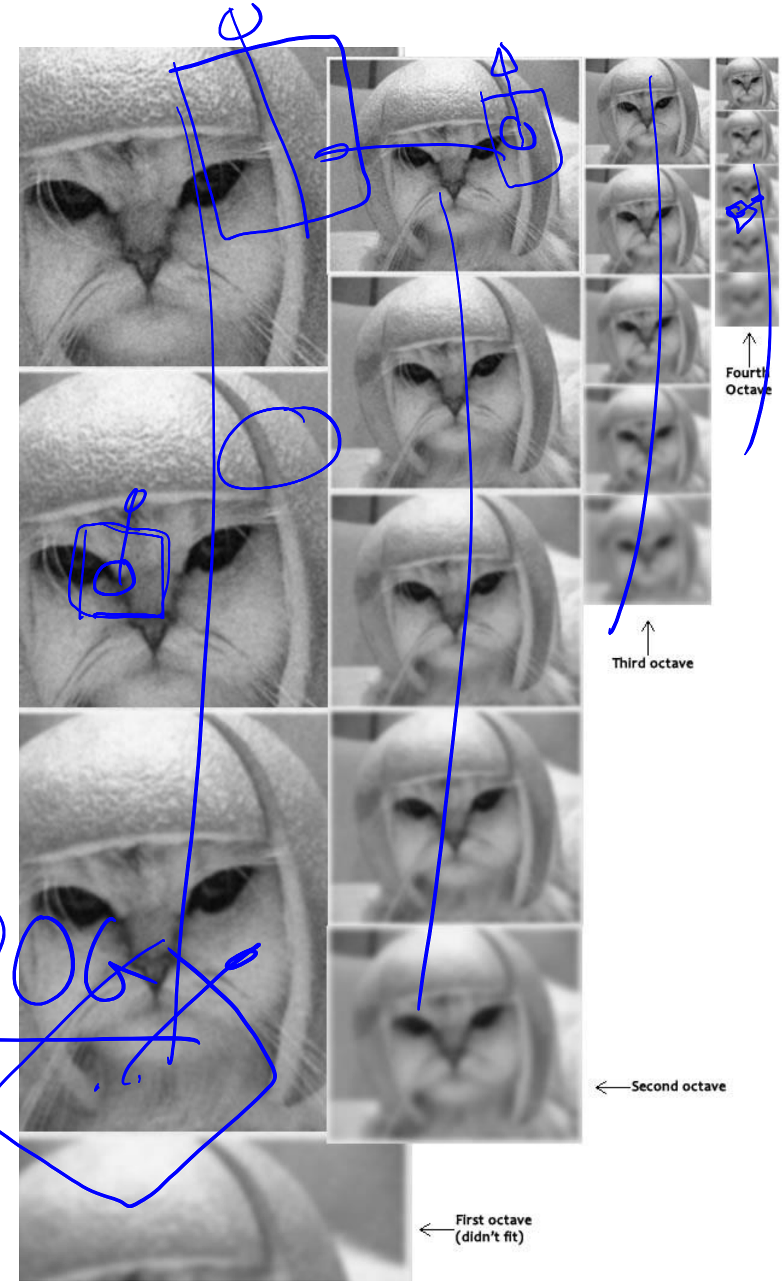
128-d   128-d



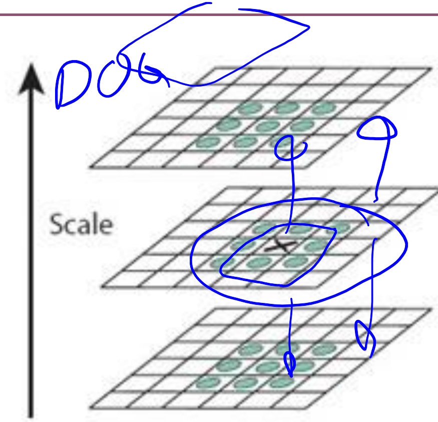
- Step 1: We create a pyramid of images using Gaussian filters at different standard deviations σ and scales. SIFT calls the different scales “octaves” as shown on the right side. Each octave is down sampled to a ¼ of the previous octave. For each octave, the image is progressively blurred (Gaussian filters with increasing σ).
  - In each octave, neighboring images are subtracted to create the difference of Gaussians (DOG) which act like edge detectors for a defined frequency band
  - The DOG image pyramid contains potential edges and point of interests. They are the local minima and maxima in the DOG.



4 DOG



– Step 2: We detect the local minima and maxima in the DOG pyramid with a one pixel neighborhood. As shown in the picture on the right, where “x” marks the current pixel, we have 8 neighbors in the same plane and 9 neighbors from each the plane above and below. If the pixel is a maxima or minima in this neighborhood, mark it as such. Otherwise dismiss the pixel.

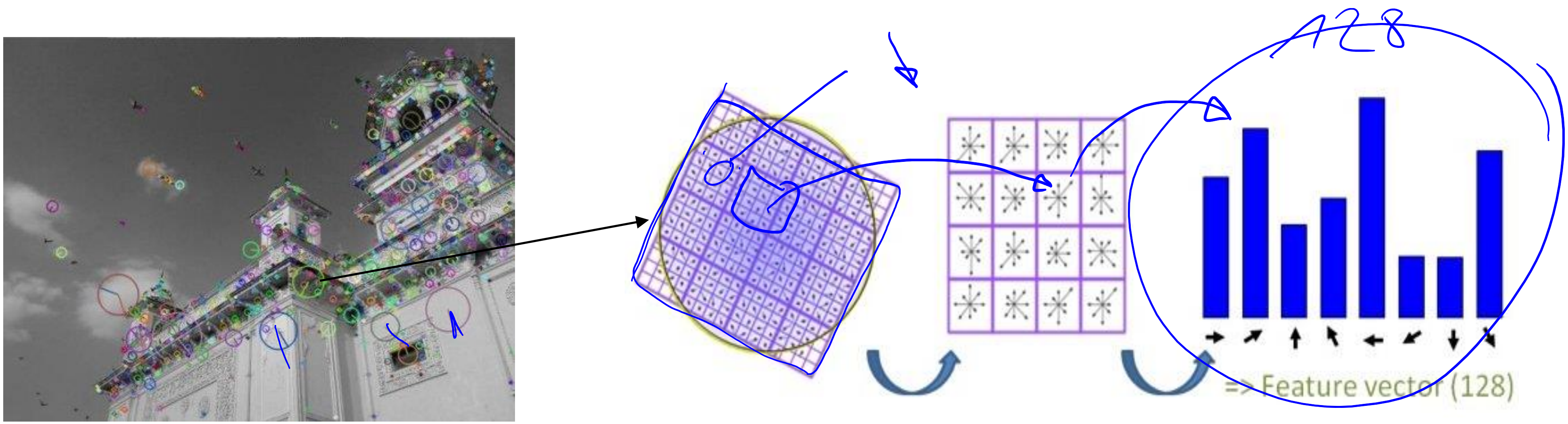


  - Starting with 5 Gaussian blurred images in each octave, we created 4 DOG images which now create 2 extrema images at each octave.
  - To thin out the number of keypoints, we dismiss all pixels whose value in the DOG is smaller than a threshold (these are points in the “flat”). We further dismiss all edges by considering their gradients. An edge has big gradient orthogonal to the edge, and a small gradient along the edge. But we are interested in corner points with two big gradients.
  - The output of step 2 is a set of keypoints with location and scale.

– Step 3: To construct a rotation invariant feature, we need to calculate a major orientation for the keypoint. SIFT accumulates a local histogram of gradient directions from the neighborhood of the keypoint. The area of the neighborhood window is proportional to the scale. A gradient direction is added to the histogram with its magnitude as the weight. Finally, the histogram bin with the highest value corresponds to the dominant direction (if there are ties, use all directions).

  - SIFT uses the dominant direction to normalize feature gathering as shown in the next step. If several directions are found, it constructs features for all directions. The normalization allows us to compare keypoints found from different viewpoints with a simple metric.
  - The dominant direction of the keypoint is not necessarily its gradient direction.

- Step 4: Using the keypoint as the center, SIFT lies a 4x4 grid in the dominant direction over the image with the size of the grid being dependent on the scale of the keypoint. For each grid cell, a finer 4x4 mesh defines its neighborhood and a histogram with 8 directions captures the directions within the cell. For each point in this finer mesh, we calculate the gradient orientation and the magnitude. We use the magnitude and a Gaussian weight (based on the distance to the keypoint) to add the direction to the histogram. For each cell of the bigger 4x4 grid we obtain a histogram with 8 values, resulting in a total of 128 feature values.
- The SIFT features are invariant to scale, translation and rotation by construction. It follows the idea of the receptive fields in the primary visual cortex to capture local features based on directions. The features are very distinct for the objects and even small objects can yield many descriptors. Although rather complex in construction, features can be obtained close to real-time. SIFT features are widely used for object recognition, motion detection, image alignment and stitching. OpenCV has a SIFT implementation, scikit-image supports similar approaches (daisy, harris). As with HOG, SIFT descriptors can be used as input for machine learning.



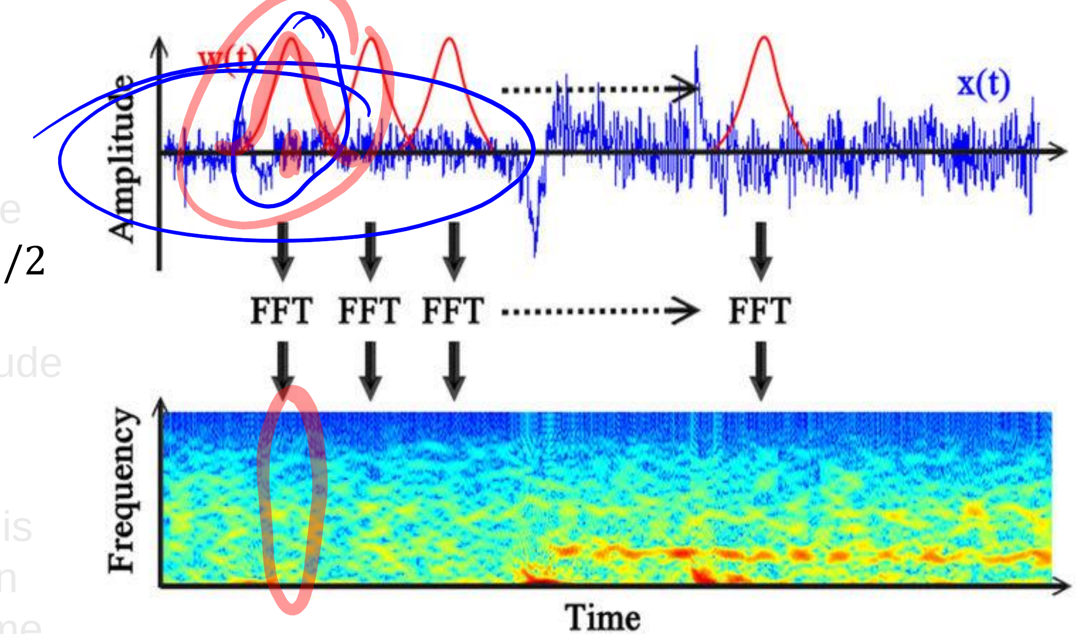
4.5 Features for Audio

- There are two definitions for sound: the first one is based on physics and describes vibrations that propagate in the form of audible pressure waves through a medium (gas, liquid, solid). The second is based on the perception through the hearing mechanism, that is, as a sensation.
- **Physics of Sound**: soundwaves are generated by a source, for instance vibrations of a speaker, and traverse a media as wave with a specific wavelength λ (or frequency f), pressure p (amplitude or intensity, measured in decibel), speed v, and direction $\vec{x}$. Note that sounds only travel if a medium exist but not in vacuum. The particles of the medium locally vibrate but do not travel with the wave.
  - The human ear perceives frequencies between 20Hz and 20kHz, corresponding to sound waves of length 17m and 17mm in air at standard conditions, respectively. The relationship between wavelength and frequency is given by the speed of the wave: $\lambda \cdot f = v$.
  - The speed of sound waves depend on the medium: in air under standard conditions, sound travels with $v = 331 + 0.6 \cdot T$ m/s with T the temperature in Celsius. In water, sound travels much faster at speeds of about $v = 1482$ m/s. In solids, speeds are even higher ranging from $v = 4000$ m/s in wood up to $v = 12{,}000$ m/s in diamonds.
  - Sound travels in concentric waves that can be reflected, refracted (when passing from one medium to another), and attenuated (gradual loss of intensity as the wave travels). With the physic properties, it is possible to locate the source of the sound (or most recent reflection point).
  - Sound pressure is the difference between the local pressure in the medium and the pressure of the wave. It is often expressed as decibel: $L_p = 20 \cdot \log_{10}(p/p_{ref})$ with p the sound pressure and p_{ref} the reference pressure (20 μPa in air). The factor 20 (and not 10) is because we compare squares of pressures; with the logarithm, this adds and extra factor of 2. The logarithmic scale is necessary due to the wide dynamic range of perception. 0 dB is the auditory threshold and sounds above 120 dB may cause permanent hearing loss.

- **Perception of Sound**: historically, the term sound referred exclusively to the auditory perception ("that which is heard"). Nowadays, the term is used both for the physical effect as well as the sensation of that effect. The perception is bound to a range of frequencies. The human ear can perceive frequencies between 20Hz to 20kHz. A cat perceives frequencies between 500Hz and 79kHz. The higher range is useful to detect high frequency mice communication (at 40kHz). Bats have a range from 1kHz up to 200kHz and use the ultrasonic sounds for echolocation of prey. The elements of sound perception are:
  - **Pitch**: is the perceived (primary) frequency of sound. It is a perceptual property that allows us to judge music as "higher" or "lower". Pitch requires a sufficiently stable and clear frequency to distinguish it from noise. It is closely related to frequency but not identical.
  - **Duration**: is the perceived time window of a sound, from the moment it is first noticed until it diminished. This is related to the physical duration of the wave signal, but compensates breaks of the signal. For instance, a broken radio signal can still be perceived as a continuous message.
  - **Loudness**: is the perceived level ("loud", "soft") of a signal. The auditory system stimulates over short time periods (~200ms): a very short sound is thus perceived softer than a longer sound with the same physical properties. Loudness perception varies with the mix of frequencies.
  - **Timbre**: is the perceived spectrum of frequencies over time. Sound sources (like guitar, rock falling, wind) have very characteristic timbres that are useful to distinguish them from each other. Timbre is a characteristic description of how sound changes over time (like a fingerprint).
  - **Sonic Texture**: describes the interaction of different sound sources like in an orchestra or when sitting in a train. The texture of a quiet market place is very distinct from the one of busy party.
  - **Spatial Location**: denotes the cognitive placement of the sound in the environment (not necessarily the true source) including the direction and distance. The combination of spatial location and timbre enables the focused attention to a single source (e.g., partner at a party).

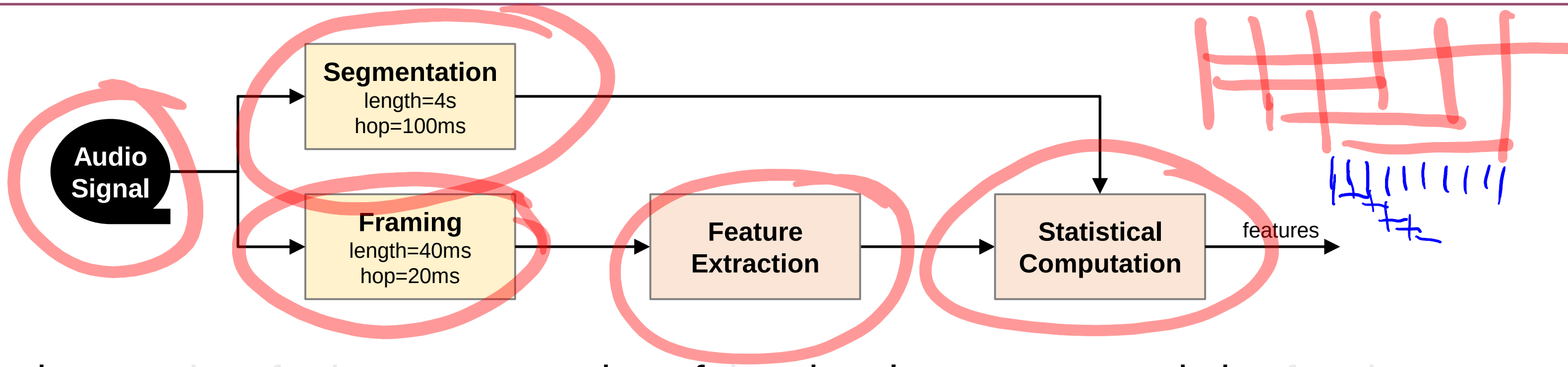
- Audio signals are expressed as an amplitude signal over time. To capture the continuous signal and create a discrete digital representation, the signal is sampled with a fixed frequency f_s. The Nyquist–Shannon sampling theorem states that the sampling rate limits the highest frequency f_{max} that can be resolved to half of the sampling rate ($f_{max} = f_s/2$). As the human perception ranges between 20Hz and 20kHz, sampling rates of CDs was defined at 44.1kHz and the one for DVD at 48kHz.
- To model human perception, it is necessary to transform the raw amplitude signal into a frequency space. Unlike with images, we cannot apply a Fourier transformation across the entire signal as this would average frequencies across the entire time scale and does not allow for an analysis of frequency changes over time. Instead, the Short-Term Fourier Transform (STFT) applies a window function and computes a local Fourier transformation around a time point and a given window size. In the discrete form the STFT of the raw amplitude signal $x(t)$ is given as:

$$X(t,\omega) = \sum_{n=-\infty}^{\infty} x(n) \cdot w(n-t) \cdot e^{-i\omega n}$$



With a window size of N samples, the discrete frequency ω ranges between 0 and $f_{max} = f_s/2$ at steps of f_s/N Hz. The absolute values of the complex value $X(t,\omega)$ denote the magnitude of the frequency ω at time point t

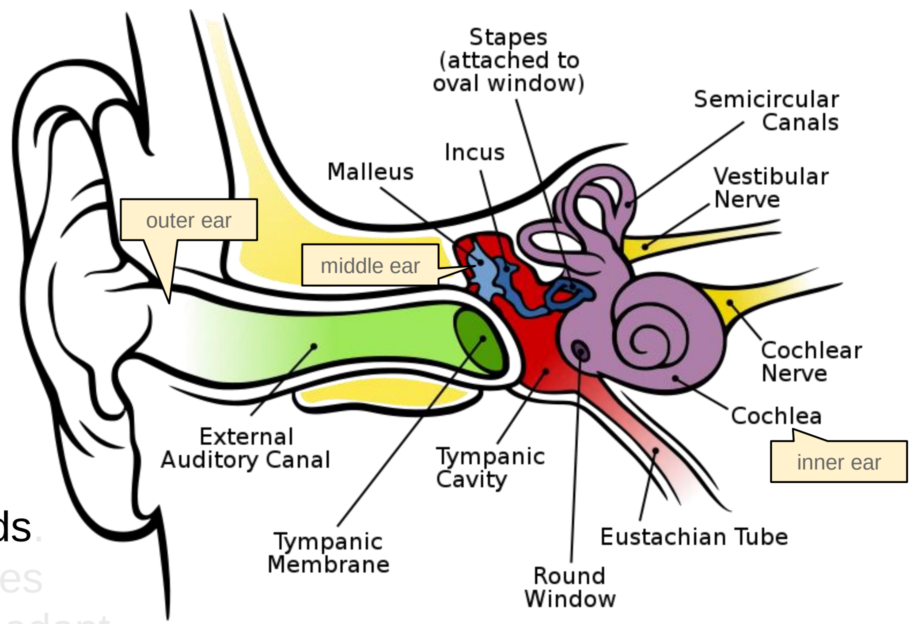
  – The picture on the right depicts the STFT with the red windowing function $w(t)$ as it is applied over time. The spectrogram is then the squared magnitudes $|X(t,\omega)|^2$ over time. One can use different windowing functions.



- Feature design requires further segmentation of the signal to capture statistics for changes over small time chunks (compare with timbre). In the picture above, the audios signal is first split into frames on which also the STFT is applied. The frames overlap with each other to avoid boundary effects. For each frame, we obtain a single feature vector. The second split of the audio signal creates overlapping segments encompassing several subsequent frames. The segment features are then a statistical summary over the features of its frames. The segments are the smallest unit for retrieval, and a single audio file is described by hundreds or thousands of segments.
  - Frame size: let the sampling frequency be $f_s = 48$ kHz. With a frame size of 40ms, the number of samples is $N = 1920$. Hence, the frequency resolution of STFT is $\frac{f_s}{N} = 20.83$ Hz. This is hardly sufficient to distinguish two musical pitches at the middle octave, but not for the first and second octave (each octave doubles the frequency). To improve frequency resolution, we could increase the window size (reducing sampling rate would result to audible artefacts). But then, we loose precision along the time axis as a broader range blurs the spectrum. In short, STFT requires us to compromise either on frequency resolution or time resolution. Alternative approaches with wavelets have solved this issue and provide both good time and frequency resolution.
  - Segment size: depends on the task at hand. For timbre detection (guitar, rock falling, wind) a shorter segment can be used. For spoken text, alternative segmentation approaches can be used. The 4s in the picture is a good starting point for generic audio analysis.

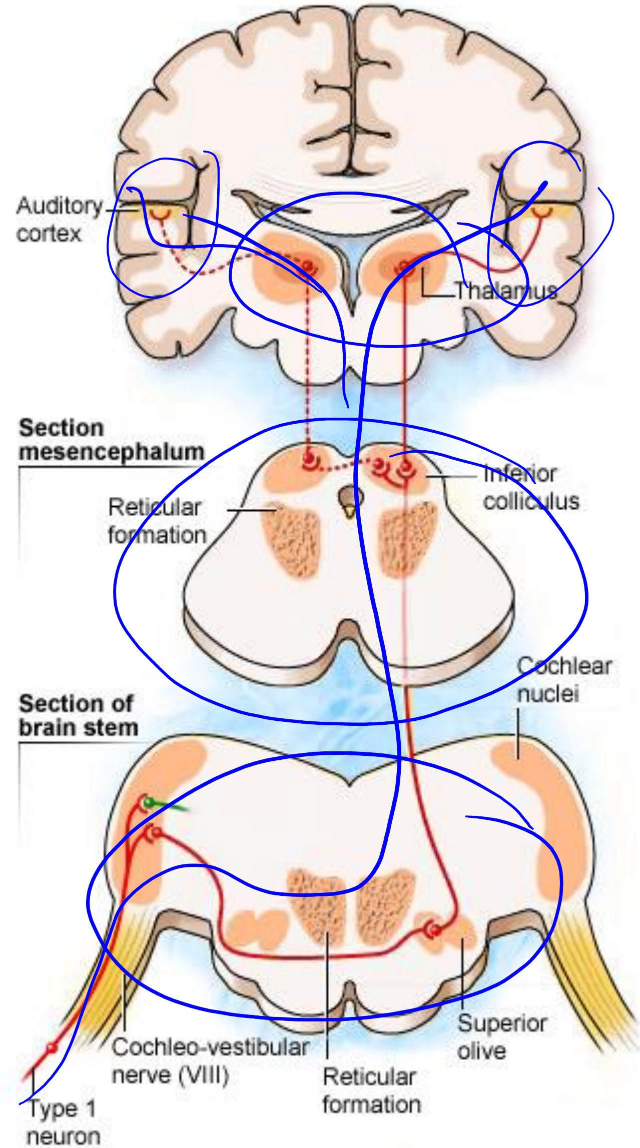
4.5.1 Auditory Perception and Processing

- The ear translates incoming pressure changes into electro-chemical impulses
  - The **outer ear** is the visible part of the organ. Sound waves are reflected and attenuated, and additional information is gained to help the brain identify the spatial location. The sound waves enter the auditory canal which amplifies sounds around 3kHz up to 100 times. This is an important range for voice recognition (e.g., to distinguish ‘s’ from ‘f’). Sound travels through the ear canal and hits the eardrum (tympanic membrane).
  - Waves from the eardrum travel through the **middle ear** (also filled with air) and a series of very small bones: hammer (malleus), anvil (incus), and stirrup (stapes). These bones act as a lever and amplify the signal at the oval window (vestibular window). Amplification is necessary as the cochlea is filled with liquid. A reflex in the middle ear prevents damage from very loud sounds.
  - The **inner ear** consists of the cochlea and the vestibular system. The latter is responsible for balance and motion detection and works similar to the cochlea. Along the cochlea runs the organ of Corti (spiral corti) with the hair cells. The outer hair cells amplify the signal and improve frequency selectivity. The inner hair cells are mechanical gates that close very rapidly under pressure (gate open means “no sound”). The base of the cochlea (closest to middle ear) captures high frequency sounds while the top captures low frequency sounds. The non-linear amplification of quiet sounds enlarges the range of sound detection. Chemical processes adapt to a constant signal focusing attention to changes.



Chittka L, Brockmann - Perception Space—The Final Frontier, A PLoS Biology Vol. 3, No. 4, e137 doi:10.1371/journal.pbio.0030137

- The electro-chemical impulses created by the inner hair cells releases neurotransmitters at the base of the cell that are captured by nerve fibers. There are 30'000 auditory fibers in each of the two cochlear nerves. Each fiber represents a particular frequency at a particular loudness level. Similarly, the vestibular nerve transmits balance and motion information. There are two pathways to the brain: the primary auditory pathway (discussed below) and the reticular pathway. The latter combines all sensory information in the brain to decide which sensory event requires highest priority by the brain. The primary path is as follows:
  - The **cochlear nuclear complex** is the first "processing unit" decoding frequency, intensity, and duration.
  - The **superior colliculus** (mesencephalum) infers spectral cues from frequency bands for sound location.
  - The **medical geniculate body** (thalamus) integrates auditory data to prepare for a motor response (e.g., vocal response)
  - Finally, the **auditory cortex** performs the basic and higher functions of hearing. Neurons are organized along frequencies. Frequency maps help to identify the source of the sound (e.g., wind). Further, it performs sound links to eliminate distortions due to reflection of waves. The auditory complex is essential to process temporal sequences of sound which are elementary for speech recognition and temporally complex sounds such as music.



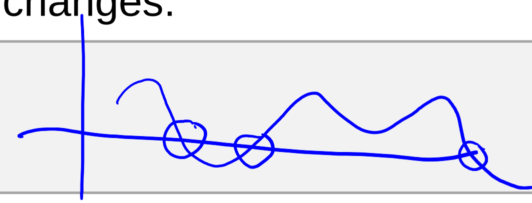
4.5.2 Generic Acoustical Features

- The first set of features describe audio files from an acoustical perspective along the domains
  - Time Domain – considering the raw signal in the time space (amplitude signal)
  - Frequency Domain – transforming raw signal with STFT and analyzing frequencies and their energies at the given time point (see window technique)
  - Perceptual Domain – modelling the perceptual interpretation of the human ear
- **Feature in the Time domain (frame)** we consider the amplitude signal in the time domain using a single frame F_i (see segmentation). For instance, with $f_s = 48$ kHz and a frame size of 40ms, the number of samples is $N = 1920$, and the hop distance between subsequent frame is 20ms.
  - **Short-Time Energy (STE)**: measures the raw energy as a sum of squares, normalized by the frame length. With audio signals, power is usually measured as decibel (which is one-tenth of a bel, a unit introduced by the first telephony system). An increase of 10 dB denotes a power change of a factor of 10. The metric is logarithmic: $L_P = 10\log_{10}(P/P_0)$. With that, STE for an amplitude signal $x(t)$ within a frame F_i (hence: $1 \le t \le N$) is defined as:

$$E_{STE}(i) = 10\log_{10}\left(\frac{1}{N}\sum_{t=1}^{N} x(t)^2\right)$$

  - **Zero-Crossing Rate (ZCR)**: counts, how often the sign of the amplitude signal over the duration of the frame F_i (e.g., from positive to negative values) changes:

$$ZCR(i) = \frac{1}{2N}\sum_{t=2}^{N} |sgn(x(t)) - sgn(x(t-1))|$$



- **Entropy of Energy (EoE)**: measures abrupt changes in the energy of the audio signal within a frame F_i. To this end, the frame is divided into L sub-frames of equal length spanning the entire frame. For each sub-frame S_l, the energy is measured and normalized by the total energy of the frame to obtain a sequence of "probabilities" that sum up to 1. The entropy of these "probabilities" is the Entropy of Energy. Choose L and N_{sub} such that $N = L \cdot N_{sub}$:

$$H_{EoE}(i) = -\sum_{l=1}^{L} e(i,l) \cdot \log_2 e(i,l) \qquad e(i,l) = \frac{\sum_{t=l\cdot N_{sub}}^{(l+1)\cdot N_{sub}-1} x(t)^2}{\sum_{t=1}^{N} x(t)^2}$$

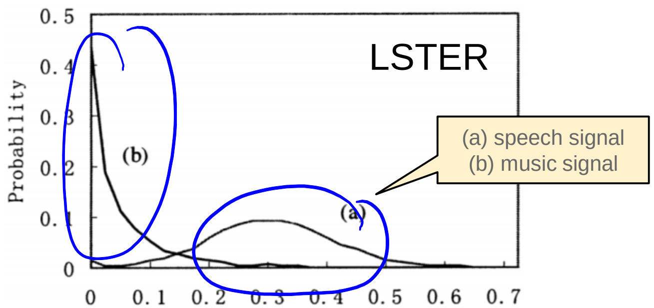
- **Feature in the Time domain (segment)**: The following features summarize statistics across a segment S_j with M frames. Consider, for instance, a segment length of 4s, a frame size of 40ms and a frame hop distance of 20ms, then the number of frames is $M = 199$ (or $M = 200$ depending on how to treat the last frame that partially is in the segment and partially outside the segment).
  - **Low Short-Time Energy Ratio (LSTER)**: denotes the percentage of frames in the segment whose STE is below a third of the average STE across the segment S_j. Speech signals have a higher variation due to pauses between syllables.

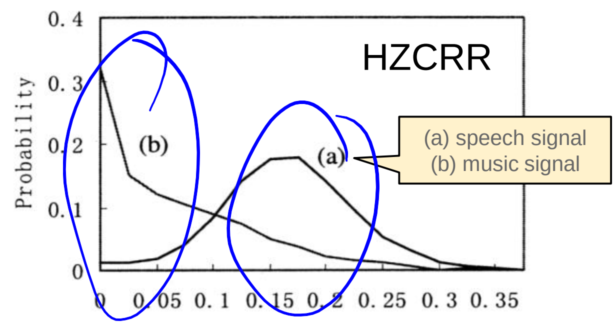
$$r_{LSTER}(j) = \frac{1}{M}\sum_{i=1}^{M}\begin{cases} 1 & E_{STE}(i) < \frac{\mu_{STE}(j)}{3} \\ 0 & \text{otherwise} \end{cases} \qquad \mu_{STE}(j) = \frac{1}{M}\sum_{i=1}^{M} E_{STE}(i)$$

- **High Zero-Crossing Rate Ratio (HZCRR)**: speech signals have much more zero-crossings than a typical music signal, and the variations is much higher (due to breaks between syllables). The HZCRR over a segment S_j is defined as:

$$r_{HZCRR}(j) = \frac{1}{M}\sum_{i=1}^{M}\begin{cases}1 & ZCR(i) > \frac{\mu_{ZCR}(j)}{3} \\ 0 & \text{otherwise}\end{cases} \qquad \mu_{ZCR}(j) = \frac{1}{M}\sum_{i=1}^{M} ZCR(i)$$

2





Lu, Zang, Content Analysis for Audio Classification Segmentation, 2002

- **Moments over STE and ZCR**: compute moments over STE and ZCR values across the segment S_j. This describes the distribution of values within the segment. The following formulas describe STE moments; ZCR moments are obtained similarly. Note that these are biased versions of the moments (which are close to unbiased moments if $M > 100$):

$$\mu_{STE}(j) = \frac{1}{M}\sum_{i=1}^{M}|E_{STE}(i)| \qquad v_{STE}(j) = \frac{1}{M}\sum_{i=1}^{M}(|E_{STE}(i)| - \mu_{STE}(j))^2$$

$$s_{STE}(j) = \frac{1}{M}\sum_{i=1}^{M}\left(\frac{|E_{STE}(i)| - \mu_{STE}(j)}{\sqrt{v_{STE}(j)}}\right)^3 \qquad k_{STE}(j) = \frac{1}{M}\sum_{i=1}^{M}\left(\frac{|E_{STE}(i)| - \mu_{STE}(j)}{\sqrt{v_{STE}(j)}}\right)^4$$

- **Histograms**: partition the value space of a feature and compute how often values fall into a partition across the frames of segment S_j. The normalized numbers yield a histogram over the feature values. This method is seldom used as it produces to large features than moments.

- **Feature in the Frequency Domain (frame)**: we consider the Fourier transformed signal in the frequency domain using a single frame F_i (see segmentation). For instance, with $f_s = 48$ kHz and a frame size of 40ms, the number of samples is $N = 1920$, and the hop distance between subsequent frame is 20ms. The Fourier transformed values $X(i,\omega)$ denotes the frequency spectrum of frame F_i with $0 \leq \omega \leq f_s/2$ and with steps $\Delta\omega = f_s/N = 25$ Hz. Also note that in the discrete notation of the Fourier transformed, i.e., $X(i,k)$ with $0 \leq k < N/2$, only the first half of the values are needed as the second half is symmetrical (as we had real values only in the time domain). In the following, we often use the discrete form $X(i,k) = X(i,\omega(k))$ with $\omega(k) = k \cdot f_s/N$.
  - **Spectral Centroid (SC)**: denotes the gravity center of the spectrum, i.e., the weighted average frequency in the spectrum of the frame F_i with the magnitude as weights, i.e., the magnitude is the absolute values of the (complex) $X(i,k)$. For convenience, let $K = N/2 - 1$. Hence:

$$SC(i) = \frac{\sum_{k=0}^{K} \omega(k) \, |X(i,k)|}{\sum_{k=0}^{K} |X(i,k)|}$$

    The centroid describes the "sharpness" of the signal in the frame. High values correspond to signals skewed at higher frequencies.

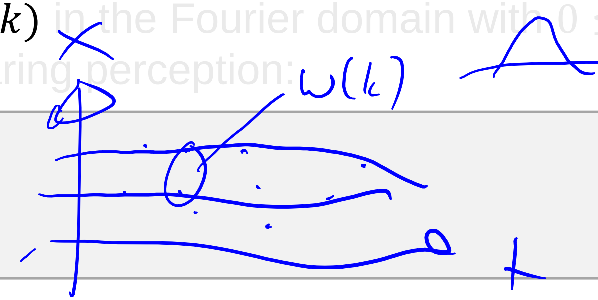
  - **Spectral Roll-off** (ω_r): denotes the frequency ω_r such that the sum of magnitudes with frequencies smaller than ω_r is $C = 85$% of the total sum of magnitudes. Hence, we look for a value $0 \leq r \leq K$ as follows (other values for $0 \leq C < 1$ are possible)

$$\omega_r = \omega(r) \quad \text{with } r \text{ smallest value that fulfills: } \sum_{k=0}^{r} |X(i,k)| \leq \mathrm{C} \cdot \sum_{k=0}^{K} |X(i,k)|$$

    Related to the spectral centroid, it measures how skewed the spectrum is towards higher frequencies which are dominant in speech.

85%

– **Band-Level Energy (BLE)**: refers to the sum of energy within a specified frequency range. The range is captured through a weighting function $w(k)$ in the Fourier domain with $0 \leq k \leq K$. The feature value is measured in decibel to match hearing perception:



$$BLE(i) = 10 \log_{10}\left(\sum_{k=0}^{K} |X(i,k)|^2 \cdot w(k)\right)$$

– **Spectral Flux (SF)**: describe the squared differences of normalized magnitudes from the previous frame. It provides information of the local spectral rate of change. A high value indicates a sudden change of magnitudes and thus a significant change of perception (only for $i > 1$):

$$SF(i) = \sum_{k=0}^{K}\left(\frac{|X(i,k)|}{\sum_{k=0}^{K}|X(i,k)|} - \frac{|X(i-1,k)|}{\sum_{k=0}^{K}|X(i-1,k)|}\right)^2$$

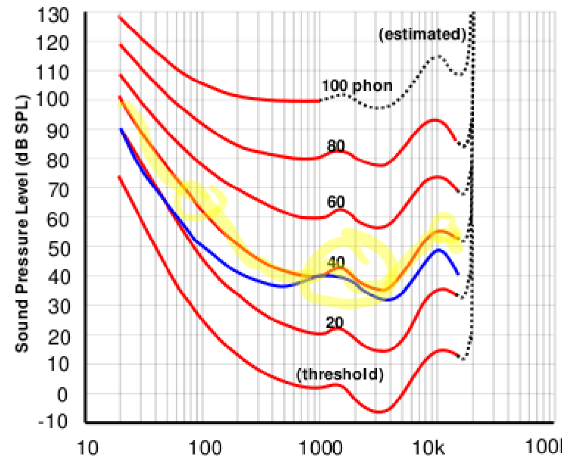
– **Spectral Bandwidth (SB)**: denotes the normalized magnitude weighted deviation from the spectral centroid. It describes the expected distance of frequencies from the spectral centroid:

$$SB(i) = \sqrt{\frac{\sum_{k=0}^{K}|X(i,k)| \cdot \left(\omega(k) - SC(i)\right)^2}{\sum_{k=0}^{K}|X(i,k)|}}$$

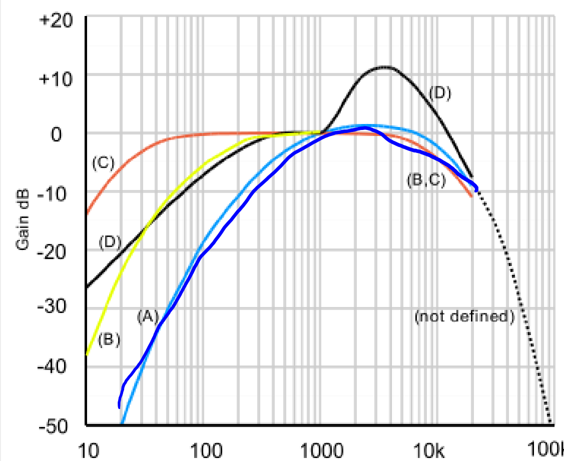
- **Feature in the Frequency Domain (segment)**: to summarize a segment, we can use again moments and histograms over the frame values for the various features above.

- **Feature in the Perceptual Domain (frame)**: the human ear and the interpretation of sound wave differs significantly from the raw physical measures. For instance, loudness is a measure of the energy in the sound wave. The human perception, however, amplifies frequencies differently, especially the ones between 2 and 5 kHz which are important for speech recognition. The following measures take perception into account.
  - **Loudness**: perception of the sound pressure level depends on the frequency as shown on the figure on the upper right side. Each red curve denotes how much energy is required such that an average listener perceives the pure tone as equally loud. As discussed before, the energy drops significantly between 2 and 5 kHz due to amplifications in the ear. To model this perception the international standard IEC 61672:2003 defined different weighting function as shown by the figure on the lower right side. The A-weighting curve is the most frequently used despite that it is only "valid" for low-level sounds. In addition, the human auditory system averages loudness over a 600-1000ms interval. The loudness at the F_i is hence the average over the previous 1000ms of the signal and not just the values in the frame. Let O be the number of frames over the last 1000ms. For instance, with a hop size of 20ms, $O = 50$. Loudness is measure in decibel, again, to match perception of increased loudness:

$$L(i) = \frac{10}{O} \sum_{o=0}^{O-1} \log_{10}\left(\frac{1}{K} \sum_{k=1}^{K} A(k) \cdot |X(i-o,k)|^2\right)$$



Equal-loudness contours (red) (from ISO 226:2003 revision)
Original ISO standard shown (blue) for 40-phons



A-weighting (blue), B (yellow), C (red), and D-weighting (blk)

- **Mel Frequency Cepstral Coefficients (MFCC)**: represents the spectrum of the power spectrum over Mel frequency bands. The Mel frequency bands approximate the human auditory system. The method works in 4 steps:

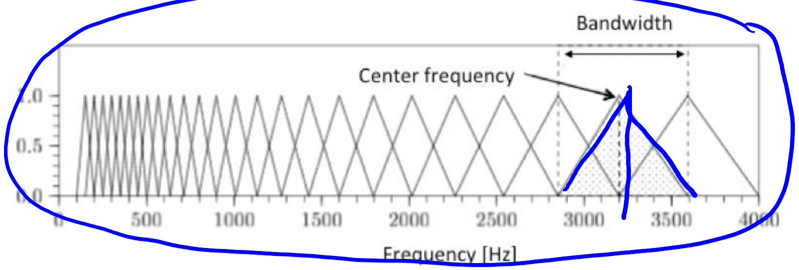
   1. Fourier Transform: compute the Fourier transform over the frame F_i. Here, we do not use a windowing function as with the STFT. Let N be the number of samples in the frame F_i and f_s be the sampling rate (e.g., $N = 1920, f_s = 48$ kHz)

$$X(t,k) = \frac{1}{N}\sum_{j=0}^{N-1} x(j) \cdot e^{-i2\pi\frac{jk}{N}} \qquad \omega(k) = k \cdot \frac{f_s}{N}$$

   2. Mel-Frequency Spectrum: the spectrum is computed over Mel frequency bands. Let B be the number of bands, and let f_{lower} and f_{upper} denote the lower and upper range of frequencies. Typically, we have $B = 26$, $f_{lower} = 300$ Hz, and $f_{upper} = 8000$ Hz. First, we create the bands. The conversion from frequencies to mels and vice versa is as follows:

$$freq(m) = 700 \cdot \left(e^{\frac{m}{1125}} - 1\right) \qquad mel(f) = 1125 \cdot \ln\left(1 + \frac{f}{700}\right)$$

The bands are triangle shaped windowing functions in the frequency space. Three frequencies define the start point, the middle point, and the end point. Two bands overlap with each other: the start point of a band is given by the middle point of the previous band. The frequencies are computed in the Mel space to match human perception. Given B bands, we need $B + 2$ frequencies given by $(0 \le b \le B + 1)$



$$f_c(b) = freq\left(mel(f_{lower}) + b \cdot \frac{mel(f_{upper}) - mel(f_{lower})}{B+1}\right)$$

With the frequencies $f_c(b)$, we can define now the windowing function $d(b,k)$ over the Fourier coefficients $X(t,k)$ for a given time point t. The shape has a triangle form:

$$d(b,k) = \begin{cases} 0 & \text{if } \omega(k) < f_c(b-1) \\ \dfrac{\omega(k) - f_c(b-1)}{f_c(b) - f_c(b-1)} & \text{if } f_c(b-1) \leq \omega(k) \leq f_c(b) \\ \dfrac{\omega(k) - f_c(b+1)}{f_c(b) - f_c(b+1)} & \text{if } f_c(b) \leq \omega(k) \leq f_c(b+1) \\ 0 & \text{if } w(k) \geq f_c(b+1) \end{cases}$$

This finally allows us to compute the Mel-frequency spectrum with a simple sum over the magnitude values of the Fourier coefficients weighted by each of the B bands. This leads to B values $M(t,b)$ for $1 \leq b \leq B$:

$$M(t,b) = \sum_{k=0}^{N/2-1} d(b,k) \cdot |X(t,k)|$$

3. Cepstral Coefficients: the cepstrum can be interpreted as a spectrum of a spectrum. The newer variant of MFCC computes the coefficients of a discrete cosine transformation and uses the first half of the coefficients. If we started with $B = 26$, we now obtain 13 cepstral values $c(t,b)$ with $1 \leq b \leq B/2$:

$$c(t,b) = \sum_{j=1}^{B} M(t,j) \cdot \cos\left(\frac{b(2j-1)\pi}{2B}\right) \qquad \text{with } 1 \leq b \leq B/2$$

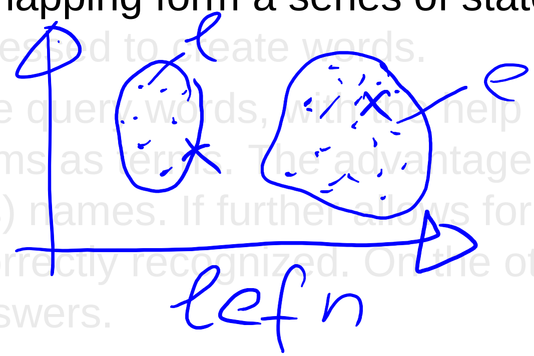
4. Derivatives: the actual MFCC features are a combination of the cepstral values $c(t,b)$ and the first and second order derivatives. The derivatives describe the dynamic nature of spoken text. With 13 cepstral coefficients, we obtain 39 feature values:

$$\Delta c(t,b) = c(t+1,b) - c(t-1,b)$$
$$\Delta\Delta c(t,b) = \Delta c(t+1,b) - \Delta c(t-1,b)$$

$$MFCC(t) = [c(t,1), \ldots, c(t,B/2), \Delta c(t,1), \ldots, \Delta c(t,B/2), \Delta\Delta c(t,1), \ldots, \Delta\Delta c(t,B/2)]$$

MFCC are the standard features for speech recognition. The feature values are used either in Hidden Markov Models or neural network to learn phonemes. A typical approach is to use the cepstral coefficients of a large spoken text body, to cluster the values into l clusters with a k-means clustering approach (see next chapter), and to use the clusters to quantize the vector and to create l states. The machine learning method then derives a mapping form a series of state transitions to a phonem. The phonem stream is then further processed to create words.

- It is also possible to search directly on the phonem stream. The query words, with the help of a dictionary, are mapped to phonems, and search is over phonems as terms. The advantage is that we do not have to train the system to recognize (countless) names. If further allows for fuzzy retrieval and is helpful if some of the phonems are not correctly recognized. On the other side, we do not have a transcript for the presentation of the answers.



- **Feature in the Perceptual Domain (segment)**: we can compute moments or histograms of the perceptual features across frames in a segment as before. The standard deviation of the 2nd MFCC coefficient $c(t,2)$, for instance, is very discriminative to distinguish speech from music.

4.5.3 Music Features (Pitch Contour)

- Chroma based features closely relate to the twelve different pitch classes from music {C, C♯, D, D♯, E ,F, F♯, G, G♯, A, A♯, B}. Each pitch class, e.g., C, stands for all possible pitches at all octaves. All pitches relate to each other by octave. If two pitches of the same class lie an octave apart, their frequency has the ratio of 1:2 (or 2:1), i.e., with each higher octave the frequency doubles. Another important concept of music theory are the partials, overtone, fundamental, and harmonics
  – Each pitched instrument produces a combination of sine waves, the so-called **partials**. The combination with its own frequencies and changes of amplitude over time define the characteristic timbre of the instrument. The human auditory system is extremely advanced to recognize timbres and to distinguish instruments (but also voices) from many audio sources.
  – The **fundamental** is the partial with the lowest frequency corresponding to the perceived pitch. **Harmonics** are a set of frequencies that are positive integer multiples of the fundamental frequency. Although an instrument may have harmonic and inharmonic partials, the design of an instrument is often such that all partials come close to harmonic frequencies.
  – **Overtone** refers to all partials excluding the fundamental. The relative strength of the overtones define the characteristic timbre of an instrument as it changes over time.

The pitch standard **A440** (also known as A4 of Stuttgart pitch) defines the A of the middle C at $f_{A4} = 440$ Hz and serves as a tuning standard for musical instruments. If we number the pitch classes (also called semitones) with $n = 0$ (C), …, $n = 11$ (B), we can express the frequency of the semitones in the octave o with $-1 \le o \le 9$ as follows (MIDI number would be 12(o + 1)+n; A4 has number 69):

$$f_{A440}(o,n) = f_{A4} \cdot 2^{\frac{12o+n-57}{12}} = 440 \cdot 2^{\frac{12o+n-57}{12}}$$

- **Table of note frequencies** (standard piano key frequencies)

Octave → Note↓	$o=-1$	$o=0$	$o=1$	$o=2$	$o=3$	$o=4$	$o=5$	$o=6$	$o=7$	$o=8$	$o=9$
C $(n=0)$	8.176	16.352	32.703	65.406	130.81	261.63	523.25	1046.5	2093.0	4186.0	8372.0
C♯ / D♭ $(n=1)$	8.662	17.324	34.648	69.296	138.59	277.18	554.37	1108.7	2217.5	4434.9	8869.8
D $(n=2)$	9.177	18.354	36.708	73.416	146.83	293.66	587.33	1174.7	2349.3	4698.6	9397.3
E♭ / D♯ $(n=3)$	9.723	19.445	38.891	77.782	155.56	311.13	622.25	1244.5	2489.0	4978.0	9956.1
E $(n=4)$	10.301	20.602	41.203	82.407	164.81	329.63	659.26	1318.5	2637.0	5274.0	10548.1
F $(n=5)$	10.914	21.827	43.654	87.307	174.61	349.23	698.46	1396.9	2793.8	5587.7	11175.3
F♯ / G♭ $(n=6)$	11.563	23.125	46.249	92.499	185.00	369.99	739.99	1480.0	2960.0	5919.9	11839.8
G $(n=7)$	12.250	24.500	48.999	97.999	196.00	392.00	783.99	1568.0	3136.0	6271.9	12543.9
A♭ / G♯ $(n=8)$	12.979	25.957	51.913	103.83	207.65	415.30	830.61	1661.2	3322.4	6644.9	
A $(n=9)$	13.750	27.500	55.000	110.00	220.00	440.00	880.00	1760.0	3520.0	7040.0	
B♭ / A♯ $(n=10)$	14.568	29.135	58.270	116.54	233.08	466.16	932.33	1864.7	3729.3	7458.6	
B $(n=11)$	15.434	30.868	61.735	123.47	246.94	493.88	987.77	1975.5	3951.1	7902.1	

Source: https://en.wikipedia.org/wiki/Scientific_pitch_notation

- Extracting pitch information from audio files requires the extraction of the fundamentals. A first, simple approach, is to map all frequencies from the STFT to a chroma value corresponding to the pitch class numbering as introduced above. We use again the A440 standard with $f_{ref} = 440$. Let $\omega(k) = k \cdot f_s/N$ be the frequency mapping of the k-th Fourier coefficient with sampling rate f_s and with N samples. Then, the chroma value (pitch class $p(k)$ and octave $o(k)$), are given as:

$$p(k) = \left\lfloor 9.5 + 12\log_2\left(\frac{k \cdot f_s}{N \cdot f_{ref}}\right)\right\rfloor \bmod 12 \qquad o(k) = \left\lfloor \frac{1}{12}\left(9.5 + 12\log_2\left(\frac{k \cdot f_s}{N \cdot f_{ref}}\right)\right)\right\rfloor$$

  - We can obtain a chroma related histogram by summing over the power spectrum using above mappings to obtain the pitch class and octave. A histogram vector for frame F_i is then:

$$h_{chroma}(i,o,p) = \frac{1}{\sum_{k=0}^{K}|X(i,k)|^2} \cdot \sum_{k=0}^{K} \begin{cases} |X(i,k)|^2 & \text{if } o = o(k) \wedge p = p(k) \\ 0 & \text{otherwise} \end{cases}$$

  - However, this does not allow to obtain the main pitch contour (or pitches if polyphonic) but simply provides a mapping to chroma values. We can estimate the fundamental f_0 in a time window if we search for the frequency which maximizes the sum of magnitudes over all its harmonics, i.e.:

$$f_0 = \frac{f_s}{N} \cdot \arg\max_k \left(\sum_{m=1}^{M} g(k,m) \cdot |X(i,km)| \right) \qquad g(k,m) = \frac{\omega(k) + 27}{\omega(km) + 320} = \frac{k\frac{f_s}{N} + 27}{km\frac{f_s}{N} + 320}$$

    $g(k,m)$ is an empirically obtained function to weight the contributions of the different harmonics. The number M is the number of considered harmonics and depends on the maximum frequency available in the spectrum.

- With the fundamental f_0, we obtain the pitch class $p(f_0)$ and the octave $o(f_0)$ of the time window. To extract several fundamentals from the frame, we repeat the following steps:
  1. Compute the magnitude spectrum $\left|X^{(0)}(i,k)\right|$
  2. Iterate $t = 0, 1, \ldots$ as long as $\sum_{k=0}^{K}\left|X^{(t)}(i,k)\right| > \epsilon$
     - Compute f_0 on the magnitude spectrum $\left|X^{(t)}(i,k)\right|$
     - Adjust the magnitude spectrum, i.e., subtract the magnitudes of the harmonics of the computed fundamental f_0 to obtain $\left|X^{(t+1)}(i,k)\right|$

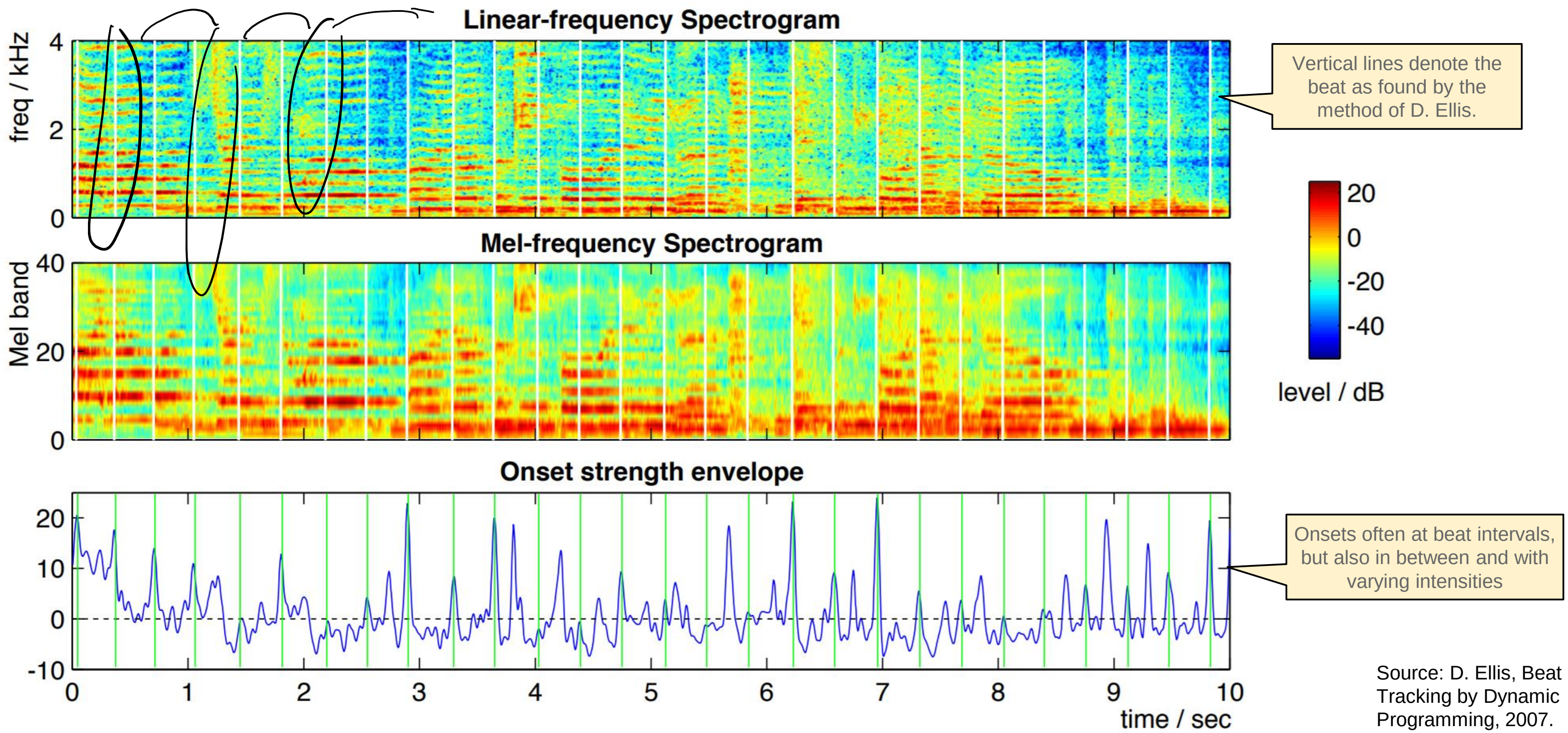
- Alternatively, we can compute the fundamental frequency f_0 in the time domain. To this end, we compute the autocorrelation of the audio signal at different time shifts Δt. Let N be the size of a frame and f_s be the sampling rate. To limit the search, we enforce the condition $1/f_{min} \geq \Delta\mathrm{t} \geq 1/f_{max}$ for a minimum and maximum frequency range for the fundamental: $f_{min} \leq f_0 \leq f_{max}$. Furthermore, time shifts are integer multiples of the sampling period: $\Delta t = m/f_s$ with $f_s/f_{min} \geq m \geq f_s/f_{max}$. The autocorrelation for the frame F_i and the lag m is then defined as follows:

$$R(i,m) = \sum_{t=m}^{N} x(i,t) \cdot x(i,t-m)$$

  To obtain the fundamental, we search for the lag m_0 that maximizes the autocorrelation and compute the frequency from this lag:

$$f_0 = \frac{f_s}{m_0} \qquad\qquad m_0 = \underset{f_s/f_{min} \geq m \geq f_s/f_{max}}{\operatorname{argmax}} R(i,m)$$

- Another music related feature is the tempo or beats per minute (bpm) of a play. In classical music, the tempo is often defined with ranges like Largo (40-60 bpm), Larghetto (60-66 bpm), Adagio (66-76 bpm), Andante (76-108 bpm), Moderato (108-120 bpm), Allegro (120-168 bpm), Presto (168-200 bpm), and Prestissimo (200+ bpm) and can vary over the play. Pop music has often a constant beat over the course of the song and bpms vary between 60 and 160, with 120 bpm being the most frequent choice for tempo.
  - Beat tracking is the search for regular onsets of energy at the beat intervals. With 100 bpm, we should observe an increase of energy at intervals of around 10ms (depending on the accuracy of the musician) indicating the beats. But it is not that straightforward as the example below shows:



Source: D. Ellis, Beat Tracking by Dynamic Programming, 2007.

– Onset envelope calculation: the onset is defined as the (positive) slope on the energy over the spectrum at a given point in time. Using STFT and mel bands, we obtain the mel spectrum $|X_{mel}(i,b)|$ as a weighted function from the frequency spectrum. The weighting is such that the areas underneath the triangular mel bands become equal. This firstly improves resolution of the lower frequencies and emphasizes them over the higher frequencies (basically a weighting by the inverse of the band width). The onset is then similar to the spectral flux, but we only consider positive slopes (hence onsets) and convert to decibels. Let B be the number of mel bans and F_i be the current frame, then the onset $o(i)$ is as follows:

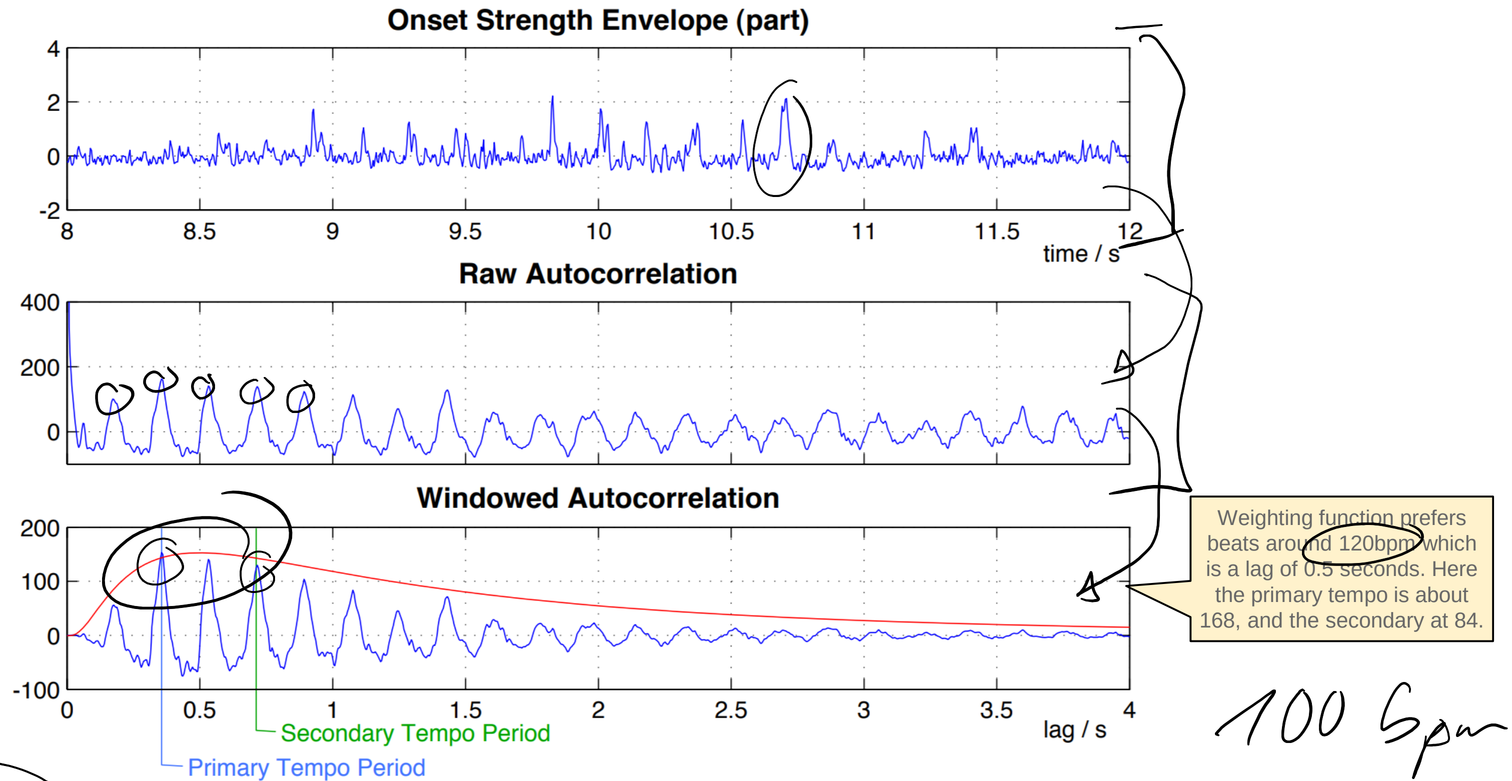
$$o(i) = \sum_{b=1}^{B} \max\left(0, \frac{\log_{10}|X_{mel}(i,b)|}{\log_{10}|X_{mel}(i-1,b)|} - 1\right)$$

– We can estimate the global tempo through autocorrelation over the onset $o(i)$ using a window function $w(i)$ (for instance using a Hann function). In other words, the we are looking for a time shift Δt such that peaks in the onset function coincide. The time shift that maximizes autocorrelation corresponds to the global tempo. We can compute the tempo per frame to obtain a tempogram, i.e., autocorrelations for the frame F_i, the lag l and the window function $w()$:

$$a(i,l) = \sum_{j=1}^{W} w(j) \cdot o(i) \cdot o(i+j)$$

The tempo is given by the lag l_0 with the highest autocorrelation and we can convert to beats per minutes with $\Delta t = l_0/f_s$ and hence the tempo is $\frac{60}{\Delta t} = 60\frac{f_s}{l_0}$ bpm. Often, we find other peaks at $\{0.33l_0,\ 0.5l_0,\ 2l_0,\ 3l_0\}$ which mark secondary tempos if their autocorrelation is large enough. In addition, we can favor beats around, for instance, 120bpms if we know the genre (e.g., pop).

Example for tempo estimation within a time frame:

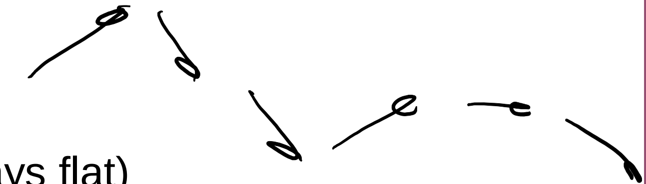


Beat tracking is then the identification of the time points $\{t_i\}$ at which the onsets occur (as a human listener would tap to the music) and such that time intervals match the tempo (with some small deviations). These time points optimize the following objective function with $F(t_i - t_{i-1}, l_0)$ being a penalty function for deviations from ideal tempo and α a weighting to balance onset values and penalty values:

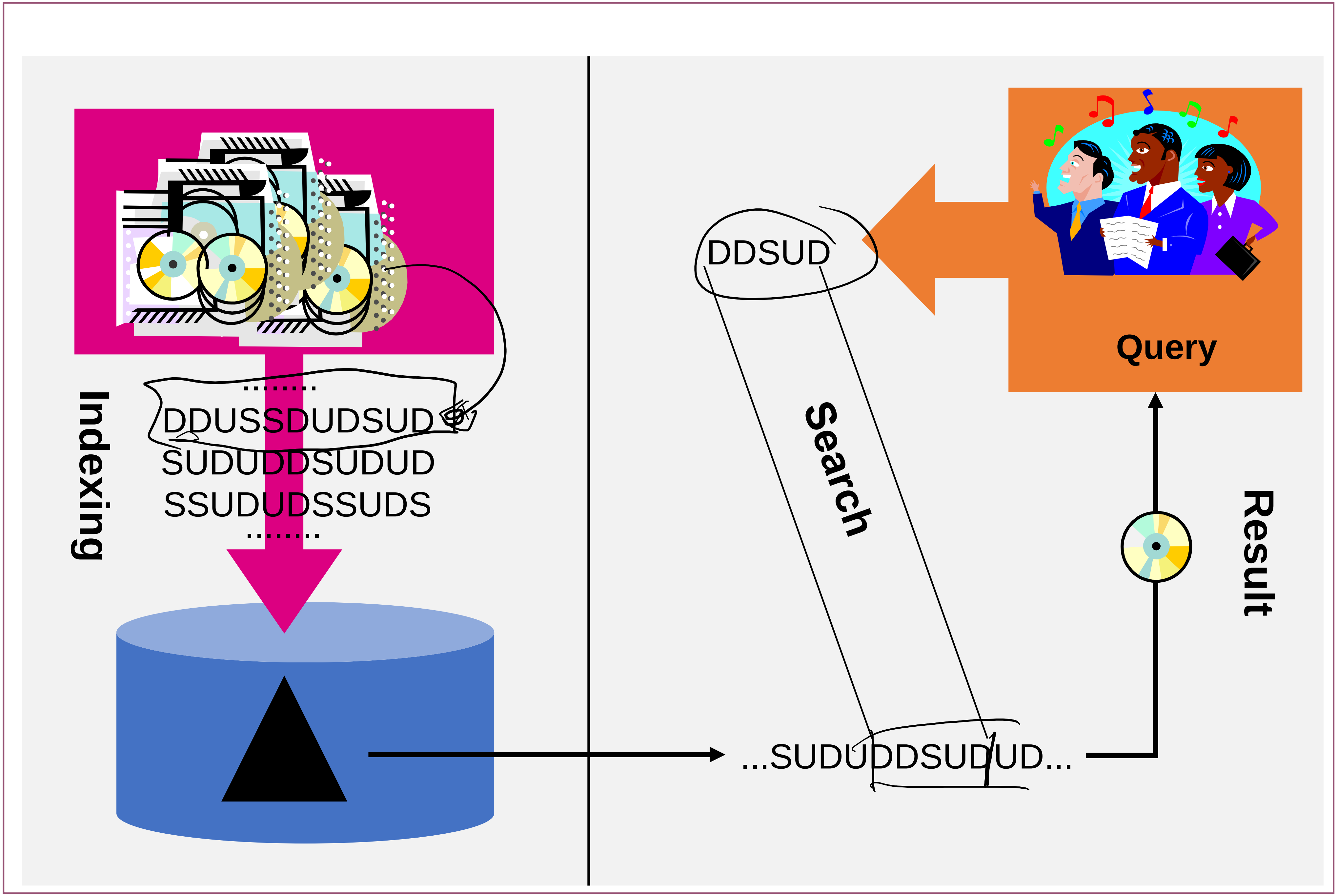
$$C(\{t_i\}) = \max_{\{t_i\}} \left(\sum_{i=1}^{T} o(f_s \cdot t_i) + \alpha \sum_{i=2}^{T} F(f_s \cdot (t_i - t_{i-1}), l_0) \right) \qquad F(\Delta l, l_0) = -\left(\log \frac{\Delta l}{l_0} \right)^2$$

4.5.4 Search for Tunes (Search by Humming)

- With music, the tune is an important piece of information. Acoustical features like beat, tempo, or pitches are not sufficient for music related search. A tune, played a different pitch levels still appears similar. A tune at a slower tempo still appears similar. Hence, we need a better way to describe a tune and to find variations of it:
  – musipedia.org is a website offering different type of tune related searches including contour search and search by humming. The idea of contour search is to describe the relative changes of the tune. For each new pitch, we note:
    - **D** (down) if the preceding pitch was higher (tune goes down)
    - **U** (up) if the preceding pitch was lower (tune goes ~~down~~)
    - **S** (same) / **R** (repeat) if the preceding pith is the same (tune stays flat)

    This transforms the stream of pitches to a stream with three terms (D, U, S). In this simple case, the duration of a pitch and pauses between pitches are ignored.
  – To search for music, one can hum the tune and the recording interface translates the humming into a sequence of terms following the above notation. The search becomes a simple string search in a database of songs.
  – There are many variations for contour search, i.e., taking duration or the step size between notes into account. Again, this translate into a contour but with additional terms. On the other side, as duration is not normalized, users may have more difficulties to hum the correct melody. The same with pitch differences: not everyone is pitch perfect but often we can remember the contour. Such interfaces are often for the more professional users.


Query
DDSUD
Search
Indexing
....
DDUSSDUDSUD
SUDUDDSUDUD
SSUDUDSSUDS
....
...SUDUDDSUDUD...
Result

4.6 Features for Video

- 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.

4.6.1 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):

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

– 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:

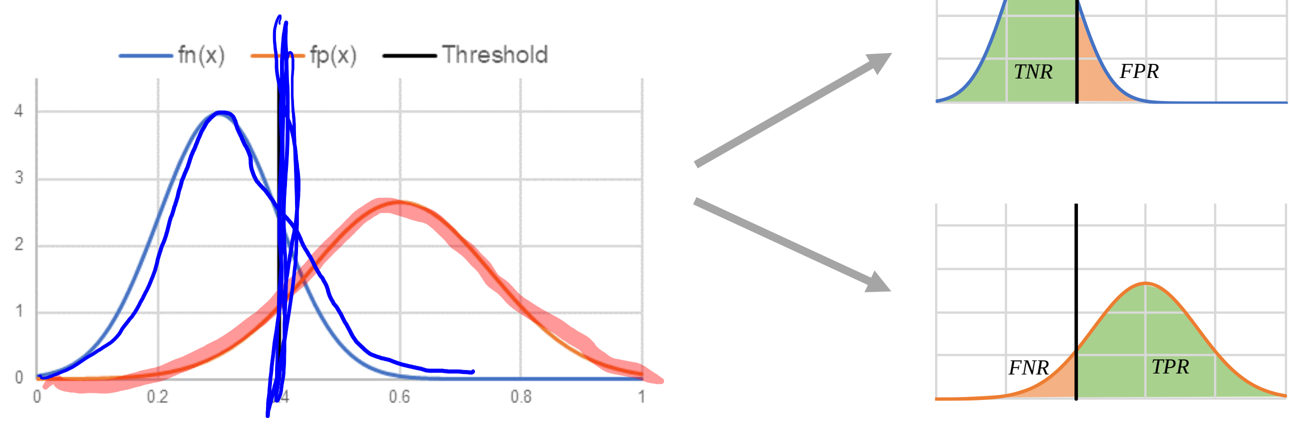
- $d_m(i) < T$ denotes that frame i belongs to the same shot (no shot boundary; negative case)
- $d_m(i) \geq T$ denotes that frame i belongs to a new shot (shot boundary; positive case)

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

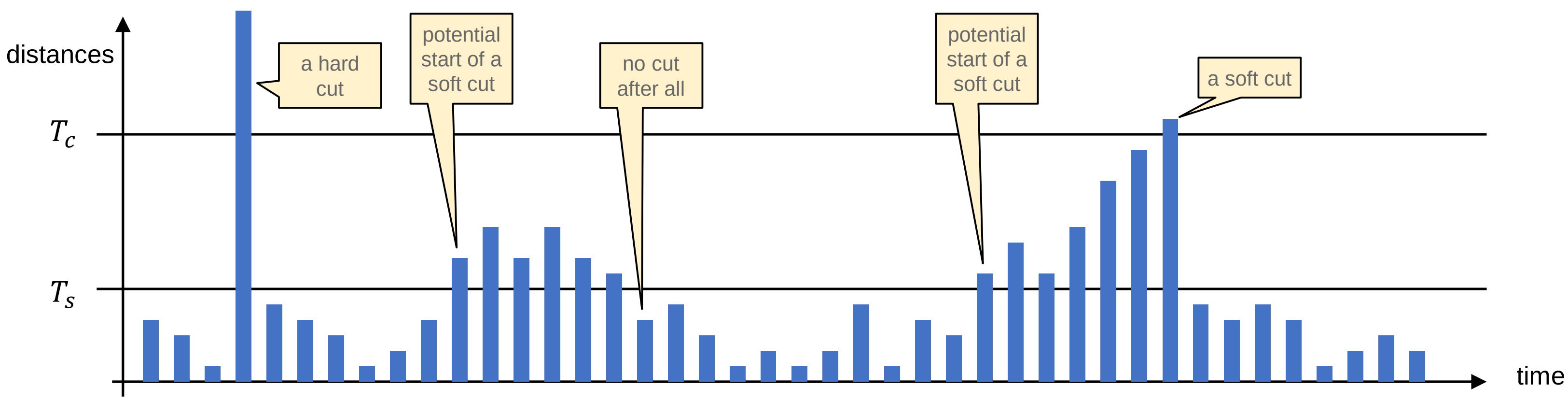
$$TPR(T) = \int_T^{\infty} f_p(x)\,dx \qquad FNR(T) = \int_{-\infty}^{T} f_p(x)\,dx$$

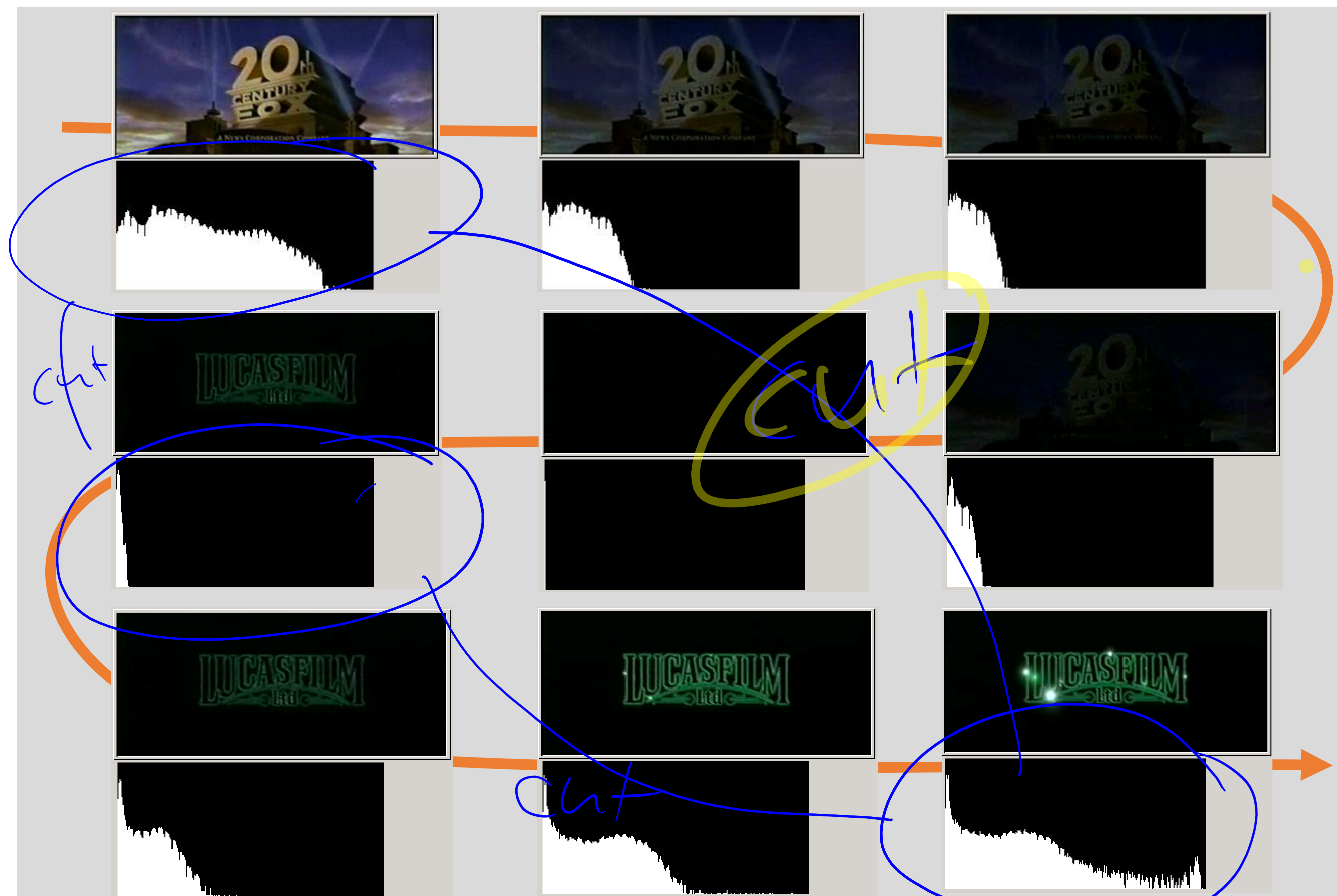
$$TNR(T) = \int_{-\infty}^{T} f_n(x)\,dx \qquad FPR(T) = \int_T^{\infty} f_n(x)\,dx$$

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.



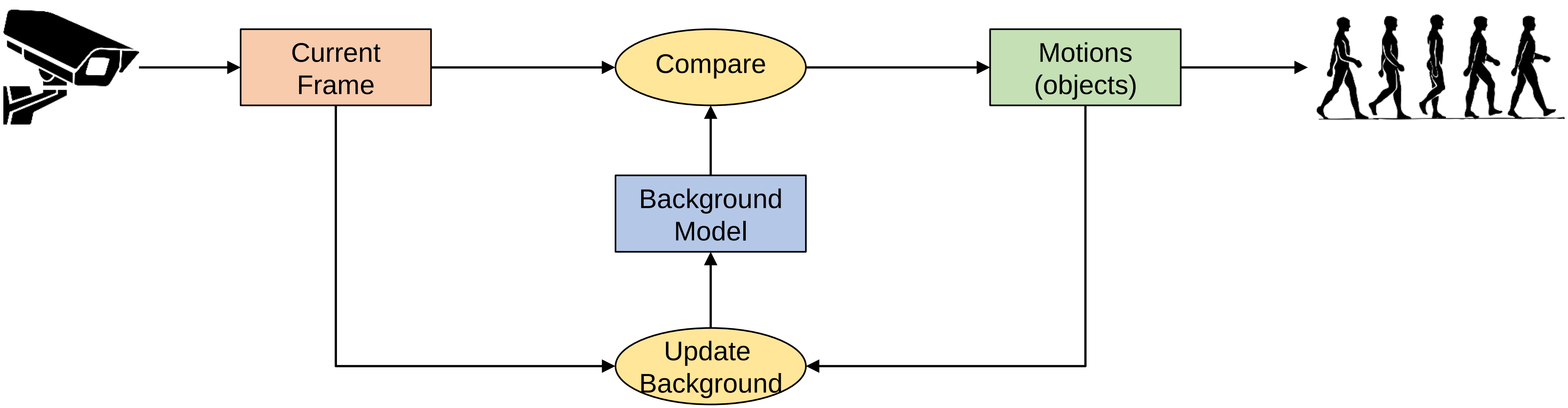
- **Shot Detection (soft cuts)**
  - The detection method above works well for hard cuts but it struggles with visual effects between shots. For instance, a slow fade-out, fade-in effect does not change subsequent images enough to trigger the threshold; but after some time, the image has changed significantly.
  - 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.



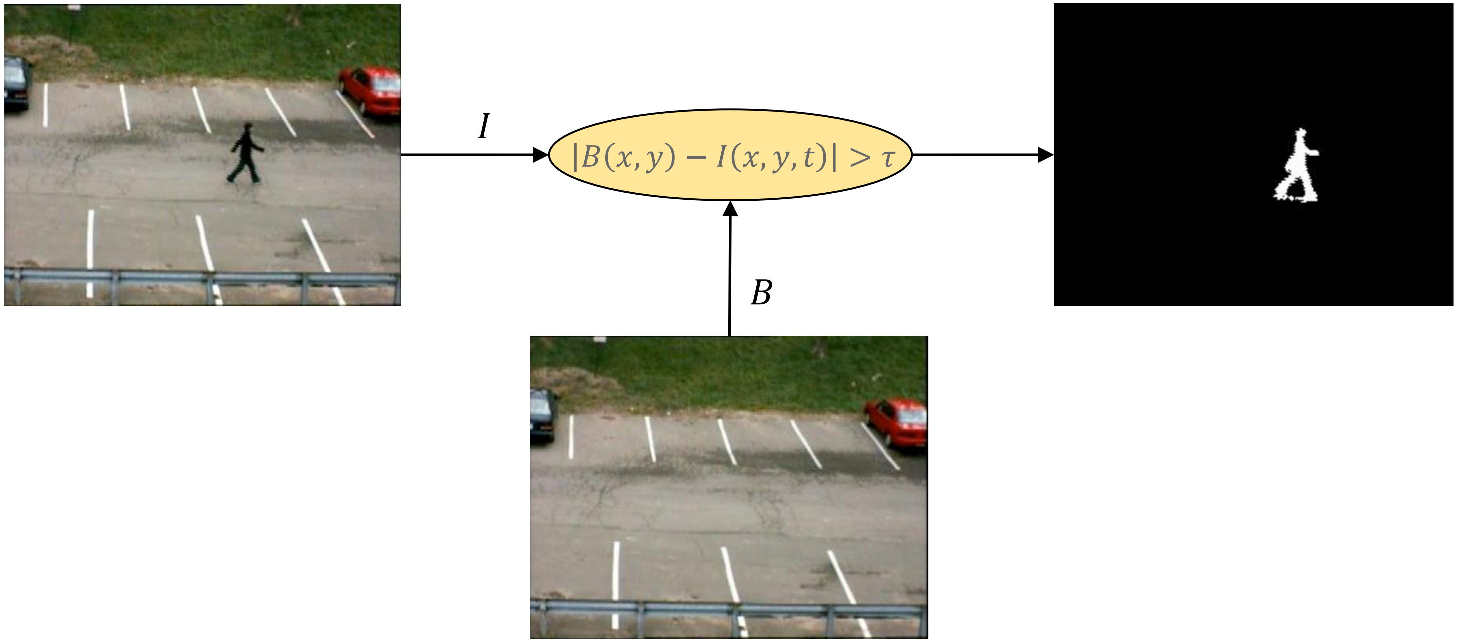

20th CENTURY FOX
LUCASFILM Ltd
cut
cut
cut

4.6.2 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



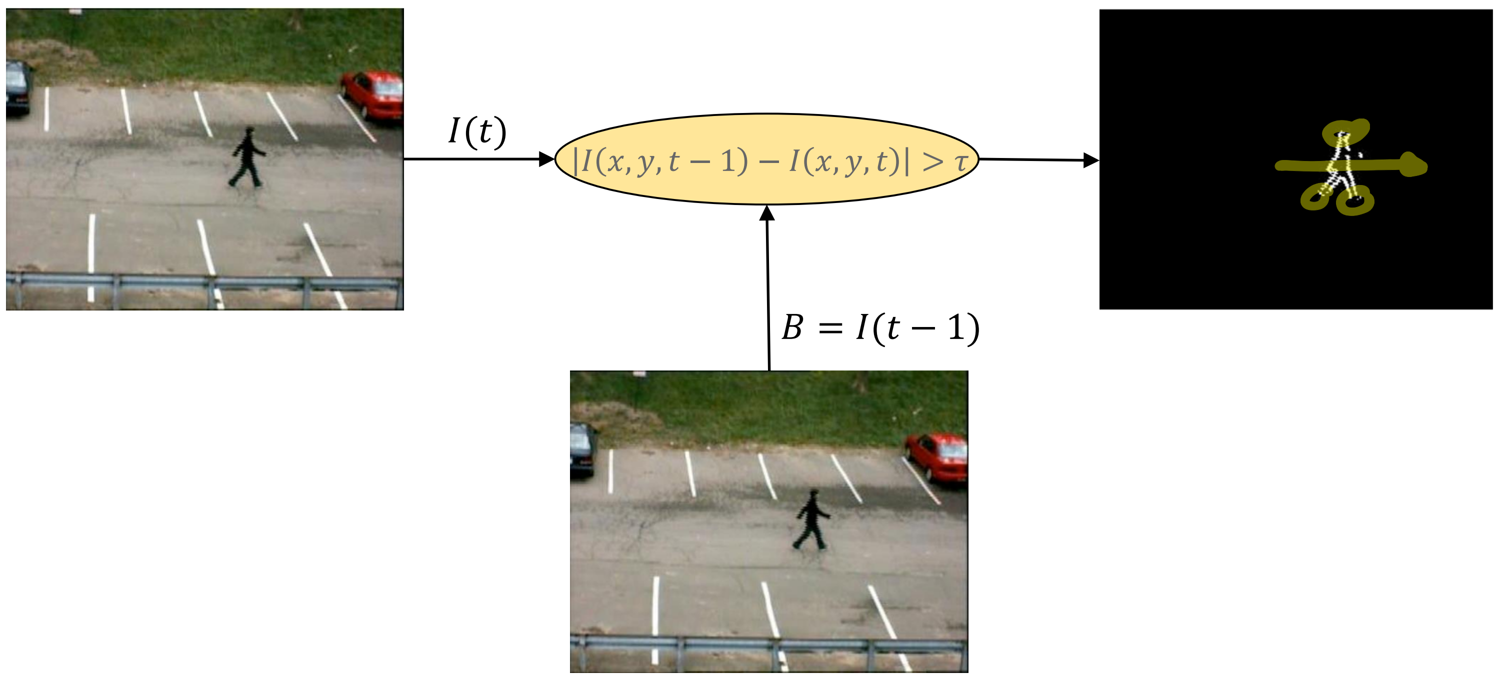
– **Simple Background Subtraction**: the background image is a static, arbitrarily selected image. We assume that the background image does not contain any moveable objects. Pixels are label with “white” if they belong to an object moving, or “black” if they are background.



- 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

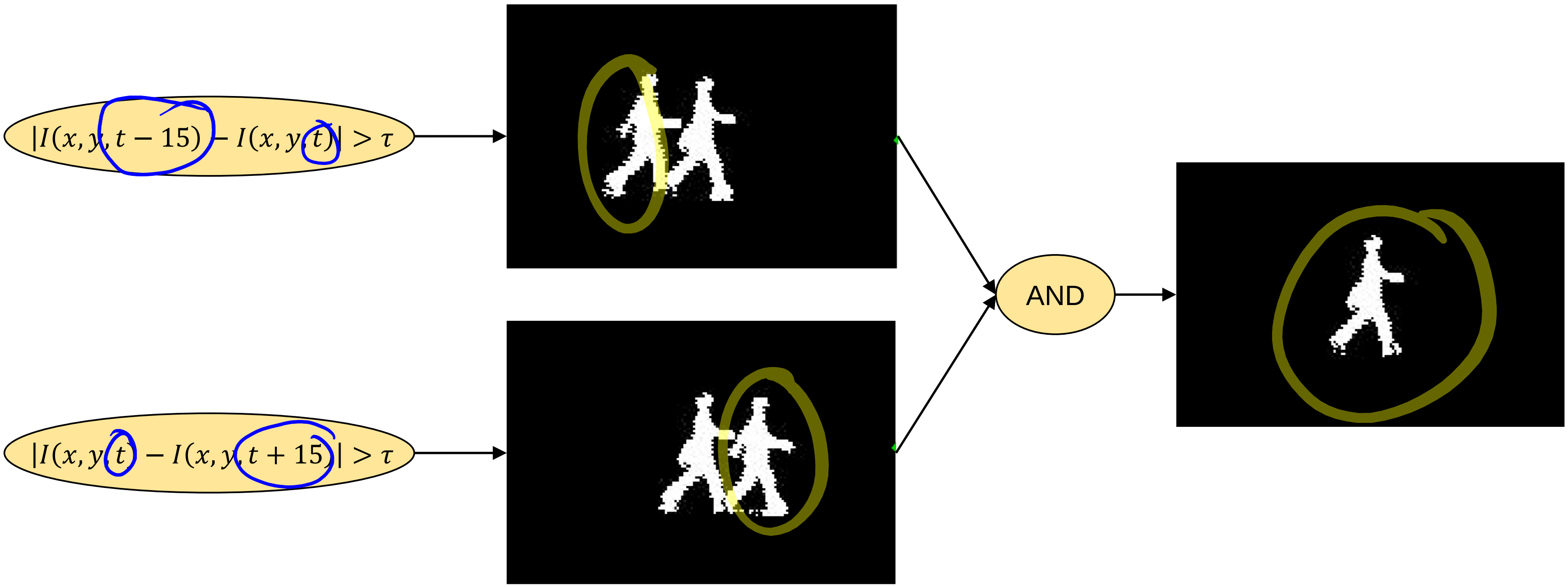
- **Simple Frame Differencing**: instead of a static background image, we build differences between subsequent frames. This allows to adjust to changes over time.



- 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

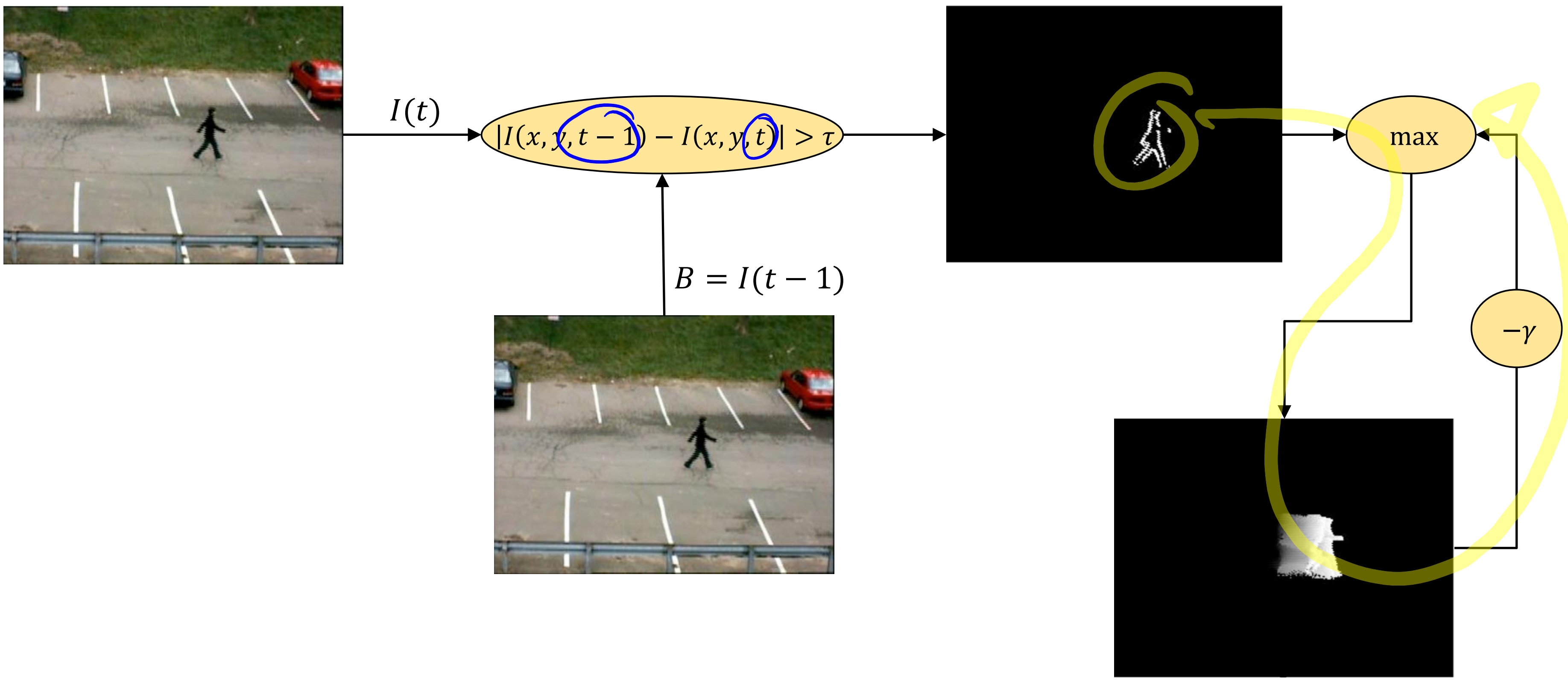
– **Three Frame Differencing**: with the simple frame differencing, we compared subsequent frames. If we enlarge the temporal distance between two frames to compare, we find more complete silhouettes but also two copies of the objects (its starting point and its current point). To eliminate copies, the 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.

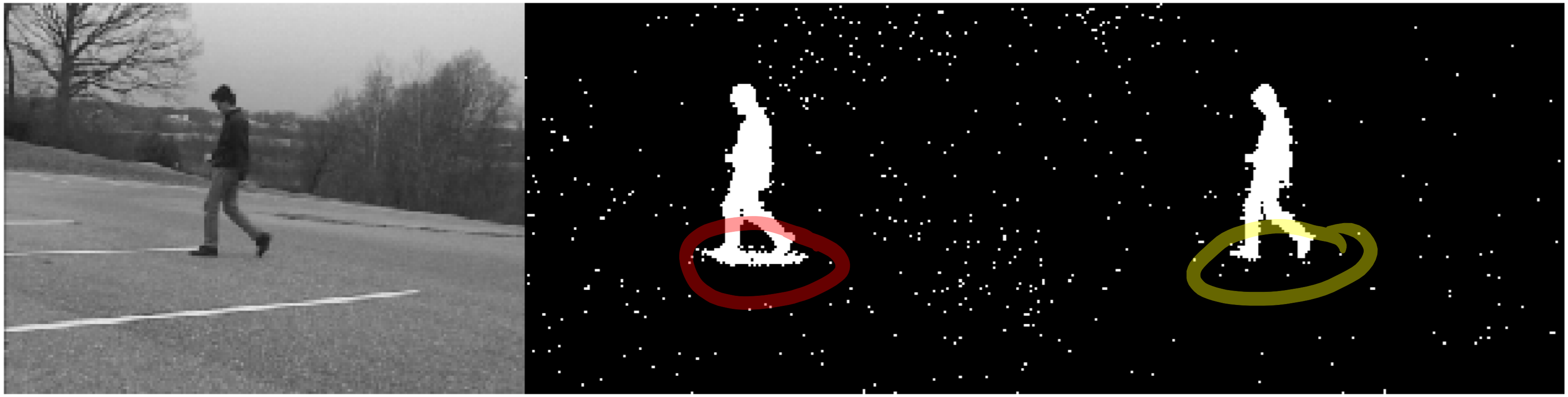


- We obtain the current motion histogram from the previous one by subtracting a chosen value γ (negative values are zeroed) and combining it (max function) with the current motion image. The decay parameter γ adds gray values to the motion history denoting how long ago the motion was detected (the darker 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:

    $I(x(t), y(t), t) = const$ for small changes of t

  - Let us track the pixel from the frame at time t to the subsequent frame at $t + \Delta t$ 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:

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

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

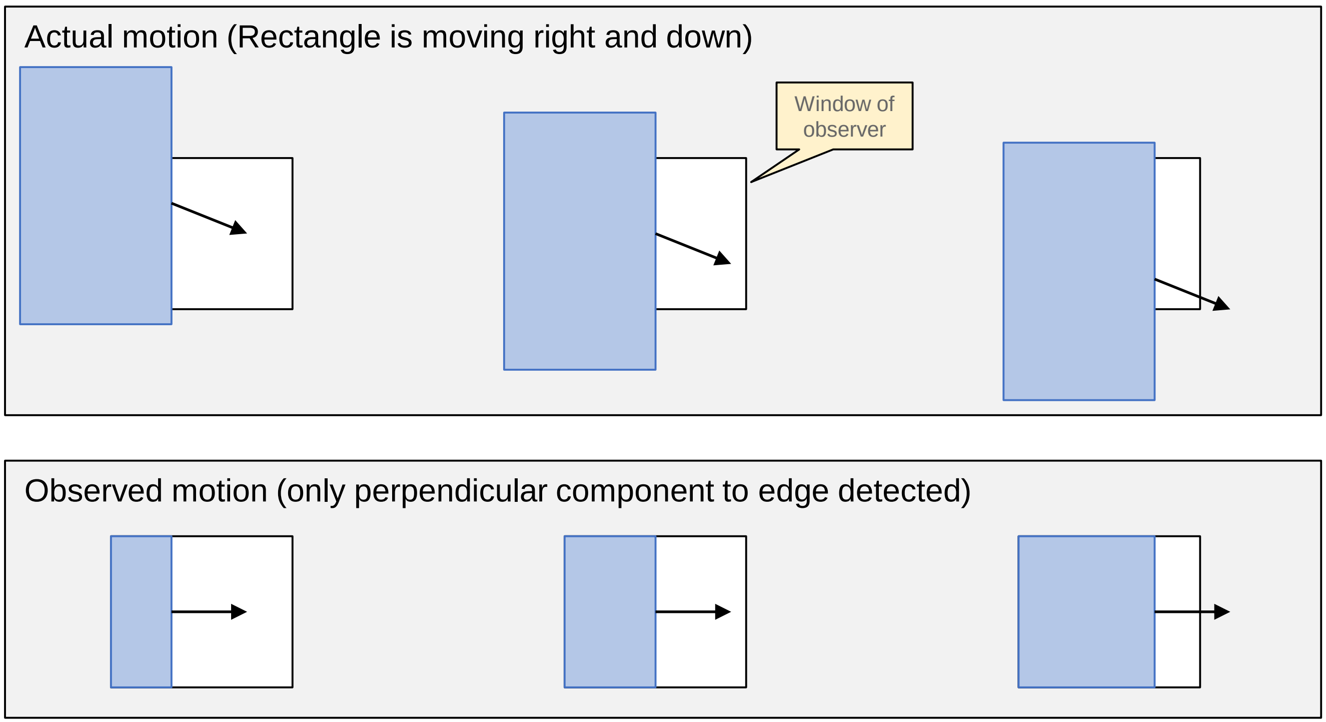
    $$0 \approx \frac{\partial I}{\partial x} u + \frac{\partial I}{\partial y} v + \frac{\partial I}{\partial t} \Delta t$$

  - 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 $t + \Delta t$.

– We obtain the final equation for our motion estimate:

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

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 $\left(I_x(x,y), I_y(x,y)\right)$. 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

– 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:

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

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

5

5

(x,y)

– 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

$$\underbrace{\begin{bmatrix} \sum I_x I_x & \sum I_x I_y \\ \sum I_x I_y & \sum I_y I_y \end{bmatrix}}_{(\mathbf{A}^\top\mathbf{A})} \underbrace{\begin{bmatrix} u \\ v \end{bmatrix}}_{\boldsymbol{d}} = \underbrace{\begin{bmatrix} -\sum I_x I_t \\ -\sum I_y I_t \end{bmatrix}}_{\mathbf{A}^\top\boldsymbol{b}}$$

$(A^T A) \cdot d = c$

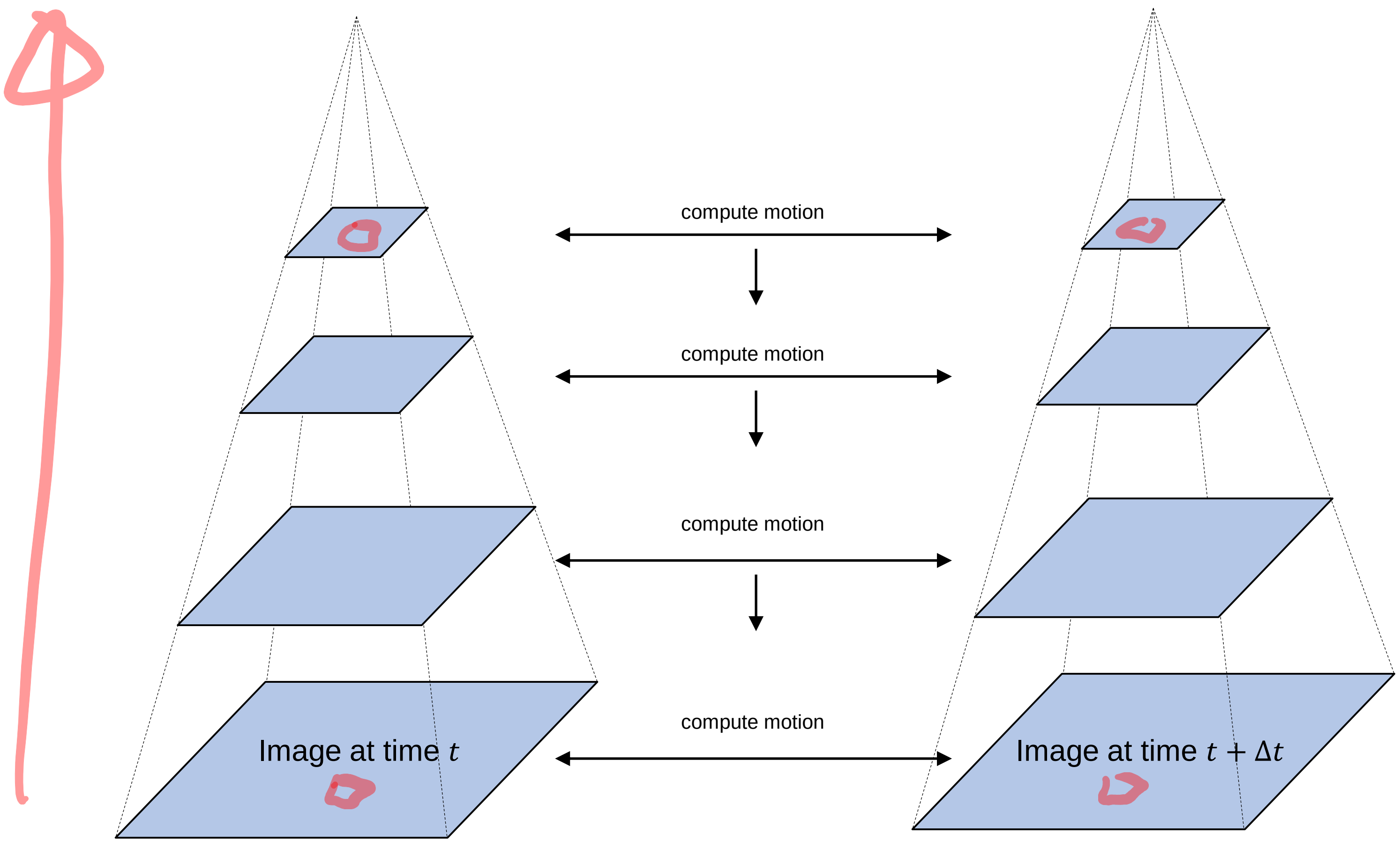
corner

edge

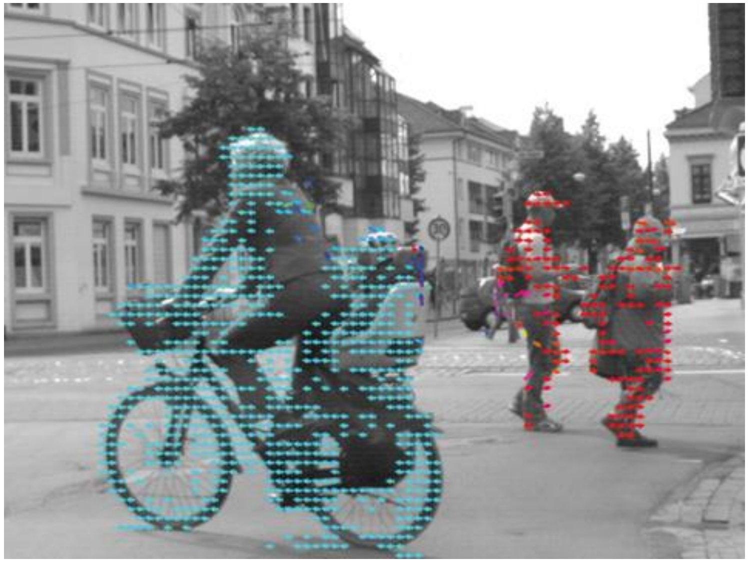
corner

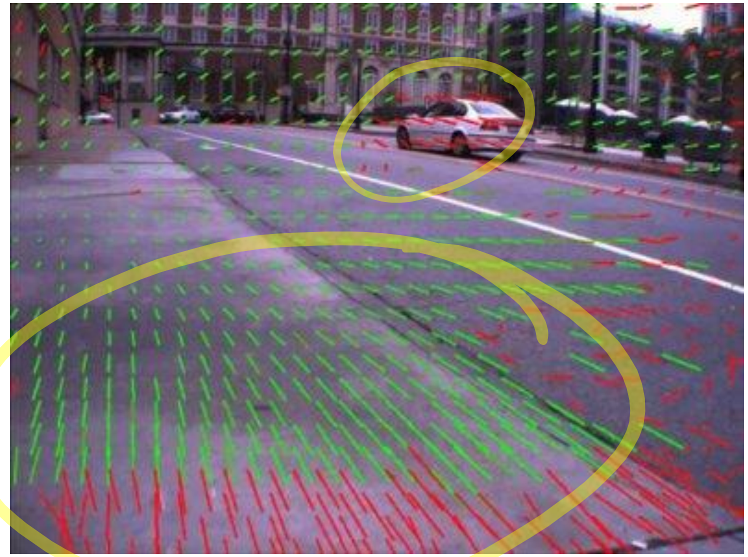
The summations are over all pixels in the 5x5 window around the current point. To solve the equation, $\mathbf{A}^\top\mathbf{A}$ should be invertible, should not have too small eigenvalues, and the ratio between the two eigenvalues should not be too large. Hence, this works best for corner points (or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

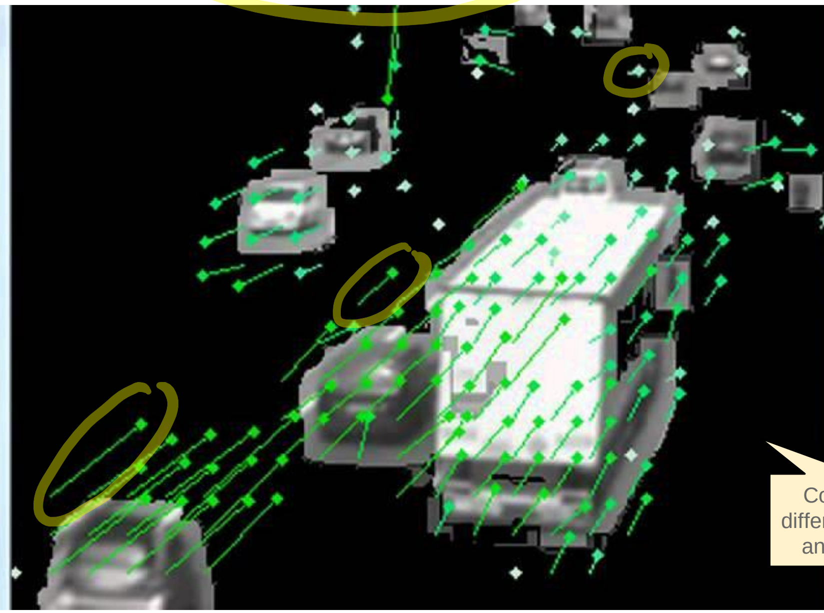
– The basic method works only well for small displacements. For large displacements, we can use a Gaussian pyramid of the image and estimate flows at each scale:



– Optical Flow: Examples (various sources)







Combination of differencing method and optical flow

4.7 Literature and Links

- Cory Doctorow, Metacrap: Putting the torch to seven straw-men of the meta-utopia, 2001, retrieved 2017 from https://www.well.com/~doctorow/metacrap.htm
- Wikipedia, List of color spaces and their uses, retrieved 2017 from https://en.wikipedia.org/wiki/List_of_color_spaces_and_their_uses
- Y. Rui, T. Huang, and S.F. Chang, **ContentBased Image Retrieval: A Survey**, Journal of Visual Communication and Image Representation, Academic Press, March 1999.
- M. Flickner, H. S. Sawhney, J. Ashley, Q. Huang, B. Dom, M. Gorkani, J. Hafner, D. Lee, D. Petkovic, D. Steele, and P. Yanker, **Query by Image and Video Content: The QBIC System.** IEEE Computer, 28(9):23-32, 1995.
- M. Stricker and A. Dimai. **Color Indexing with Weak Spatial Constraints.** In Storage and Retrieval for Image and Video Databases (SPIE), volume 2670 of SPIE Proceedings, San Diego/La Jolla, CA, USA, Feb. 1996.
- B.S. Manjunath and W.Y. Ma, **Texture Features for Browsing and Retrieval of Image Data**, IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 18:8, p. 837-842, 1996.
- A. Vailaya, **Shape-Based Image Retrieval**, Master Thesis, Michigan State University, 1996.
- S. Berchtold, D. A. Keim, and H.-P. Kriegel. **Section Coding: Ein Verfahren zur Ähnlichkeitssuche in CAD-Datenbanken.** In Datenbanksysteme in Büro, Technik und Wissenschaft, Ulm, Germany, Mar. 1997. Springer.
- A. Chalechale and A. Mertins. **Angular Radial Partitioning for Edge Image Description**. In Proc. 7th International Symposium on DSP for Communication Systems (DSPCS03), Coolangatta, Australia, Dec. 2003.
- Navneet Dalal and Bill Triggs, **Histograms of Oriented Gradients for Human Detection,** http://lear.inrialpes.fr/people/triggs/pubs/Dalal-cvpr05.pdf
- Lowe, David G. (1999). "Object recognition from local scale-invariant features". Proceedings of the International Conference on Computer Vision. 2. pp. 1150–1157. doi:10.1109/ICCV.1999.790410
- P. Mermelstein (1976), "Distance measures for speech recognition, psychological and instrumental," in Pattern Recognition and Artificial Intelligence, C. H. Chen, Ed., pp. 374–388. Academic, New York.
- 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
- R. Szeliski, **Computer Vision: Algorithms and Applications**, Textbook, http://szeliski.org/Book/

- Frameworks and Libraries
  - **OpenCV** (https://opencv.org) is an advanced computer vision library original written for C/C++. But there are also bindings for Python, Java, and other languages.
  - **scikit-image** (http://scikit-image.org) is an advanced computer vision library written in Python. It provides all basic image manipulation operations as well as advanced feature extraction algorithms (however, not SIFT but alternative approaches to SIFT)
  - **Librosa** (http://librosa.github.io/librosa/) is a Python library for advances audi and music analysis. It provides base algorithms to create music retrieval systems.
  - **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/
  - Music Information Retrieval, Vienna University of Technology, Austria, http://www.ifs.tuwien.ac.at/mir/
  - Music Information Retrieval, New York University, USA, http://www.nyu.edu/classes/bello/MIR.html
  - Music Signal Processing, Columbia University, USA https://www.ee.columbia.edu/~dpwe/e4896/index.html
  - 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/