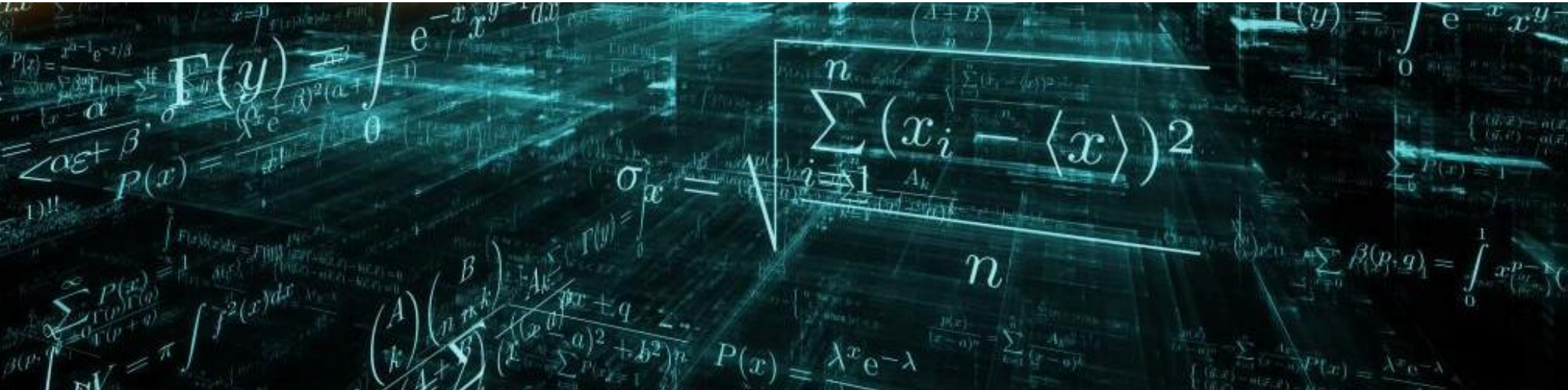
Universität Basel

Computer Science / 15731-01 / 2025

Multimedia Retrieval

Chapter 11: Visual Features

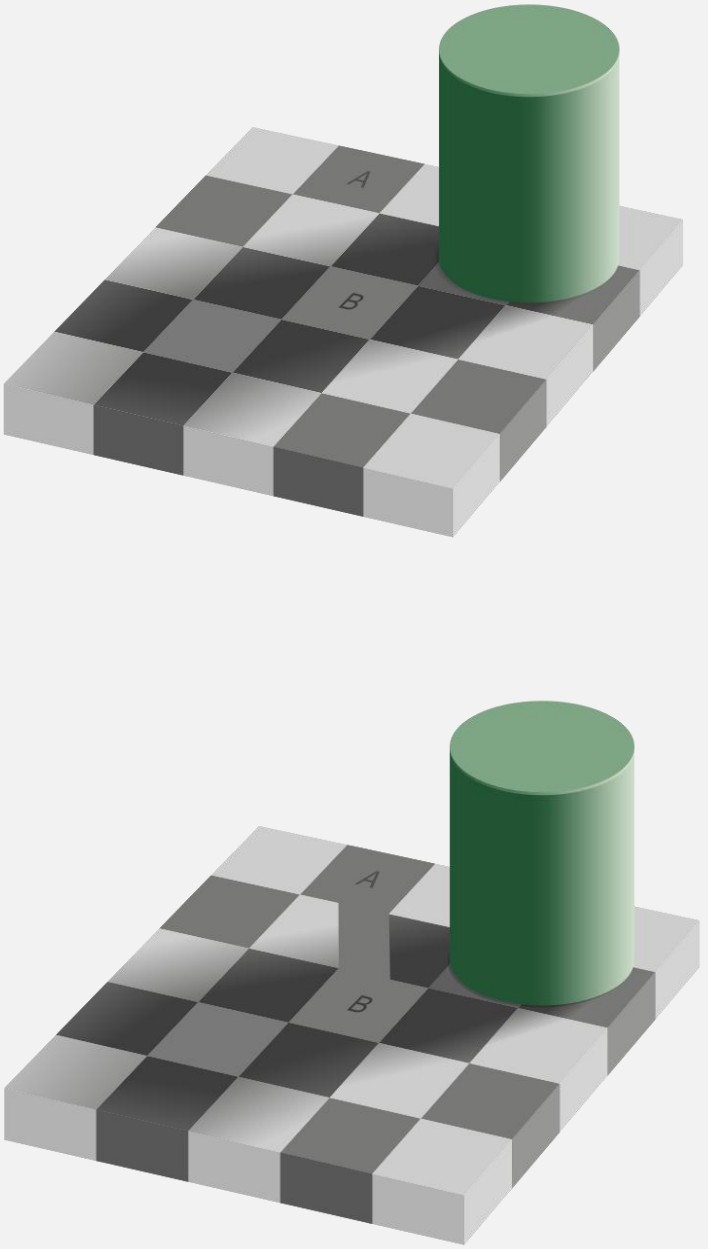
Dr. Roger Weber, roger.weber@gmail.com



11.1 Introduction

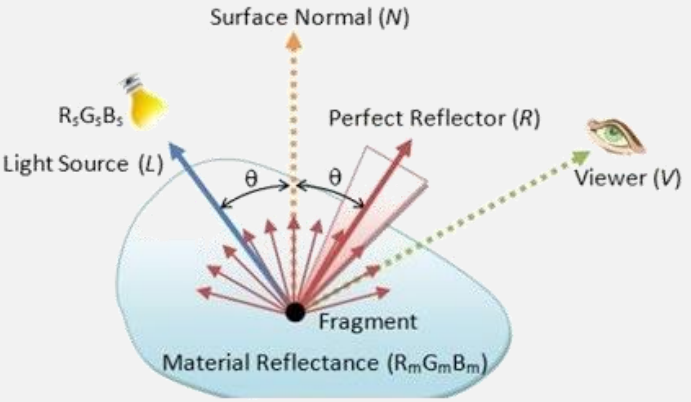
- Information retrieval research often begins with text, where meaning is studied using language models, vector embeddings, and relevance feedback. Text already has built-in meaning: words and sentences link symbols to concepts. Even when ambiguity appears, the connection between representation and meaning is clear enough for retrieval systems to form more advanced interpretations using embeddings and retrieval augmented generation. Text supports reasoning about ideas, relationships, and discourse.
- Images do not contain linguistic cues. Moving from text retrieval to visual retrieval therefore changes both the data and the gap that must be bridged. A document has words and grammar, while an image is an arrangement of light intensities and colors. The so-called semantic gap comes from the distance between that raw visual signal and the concepts people use to describe what they see. A retrieval system may be asked to find sunsets, happy faces, or industrial cities at night, yet none of those categories appears explicitly in the pixel grid.
- Before addressing this gap with deep networks or multimodal language models, it helps to review how visual representation developed in earlier computer vision. The first steps did not try to recognize objects or interpret scenes. They aimed to describe images at the level of perception rather than meaning. These methods treated an image as a visual signal whose measurable attributes reveal structure, variation and distribution. Researchers examined the role of light, the behavior of reflectance, and the constraints of the human visual system. The transformation from illumination and material to retinal response guided how color spaces are defined, how edges appear, how textures emerge, and how these properties can be summarized for retrieval.
- This approach fits into the course's broader arc. Text retrieval introduced the idea that documents can be converted to feature vectors, either by bag-of-words counts or by embeddings from large models. Moving to image retrieval keeps the same idea but at a lower level. Instead of vectors that capture meaning, we use vectors that record perceptual attributes. Color histograms, texture energies, and local gradients play the role of term frequencies, but they do not refer to high-level concepts.They are efforts to represent the visual world so it stays stable under common variations and gives enough information to support search.
- A system that can describe an image directly from its content, even at a basic level, reduces the need for external labels. It also gives an initial view of how computer systems see the world. As the course continues, these perceptual methods form the basis for later techniques, including machine learning models for classification and clustering and the CLIP model that connects images and text.

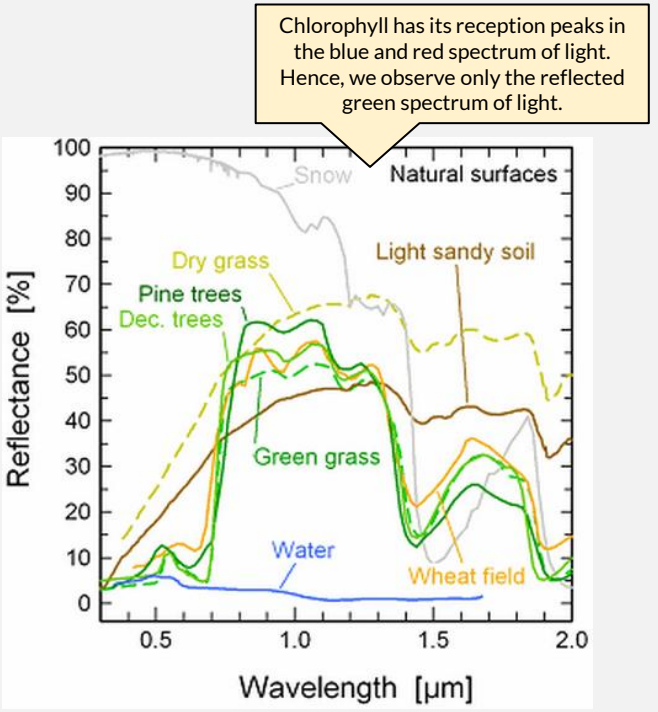
- **Visual features** are hand-crafted descriptors based on human visual processing and help close the semantic gap in image retrieval. They measure numeric properties of raw pixels to mimic how people perceive color, texture, shape, and layout. By modeling these perception steps, visual features enable content-based image retrieval, or CBIR, where queries are images, sketches, or color palettes rather than keywords.
- These methods are **inspired by the human visual system**. Light reflected from surfaces reaches the retina, where cone cells compress rich spectral information into three signals tied to long, medium, and short wavelengths. This trichromatic encoding underlies color spaces such as RGB and CIE Lab. The mapping from physical wavelengths to perceived color is not uniform. Small changes in RGB can cause large shifts in appearance, while large changes in other areas of RGB may hardly be noticed. The raw signal records physical properties, but perception follows a different interpretation. Perceptual features therefore do not try to copy the physical world. They encode how the world looks to an observer by transforming pixel values into models that measure perceptual rather than physical distance.
- Consider the two images on the right as a demonstration of how perception differs from raw measurement. The top image shows a board of alternating dark and light squares, with part of the pattern in shadow. Two squares labeled A and B look different in brightness. In the lower image a strip of the same color connects the two squares, making it clear they are the same shade. The brain interprets the scene using expectations about light and shadow rather than the raw signal. This illusion explains why **what we perceive can differ from physical measurements** and why image representations that use only pixel values do not match human judgments of similarity.



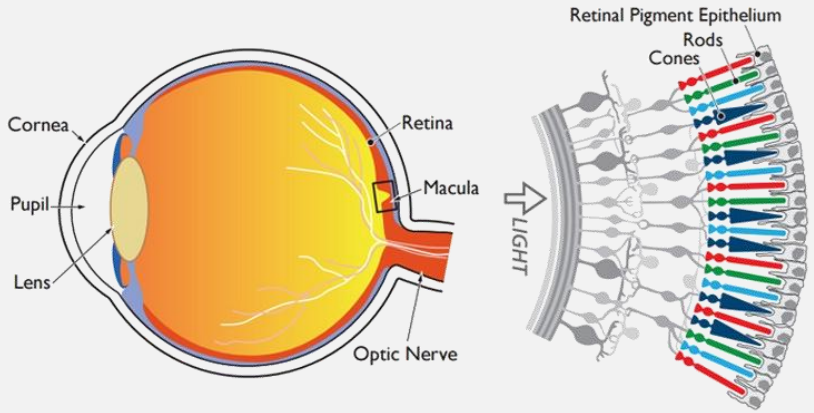
- **Designing features for perceptual images** requires understanding not only which visual elements matter but also which variations do not matter. Human perception tolerates many changes in raw pixel values as long as the meaningful content stays the same. Effective feature design aims to give computational systems the same resilience:
  - **Translation invariance** is the idea that small shifts in an image rarely change how people interpret it. For example, a photograph of a cat is still recognizable whether the cat is centered or slightly to one side. Features should be built so that minor shifts do not change their representation. Convolutional filters, pooling operations, and overlapping receptive fields are common ways to achieve this, similar to the spatial stability seen in biological vision.
  - **Rotation invariance** is also important. An object rotated by a small angle should still be seen as the same object. A coffee mug turned slightly to the side does not lose its identity, yet a naive pixel based system might treat the rotated version as completely different. Using rotation aware filters or adding rotated examples to the training data helps keep recognition consistent.
  - **Scale invariance** deals with a related problem. Objects can appear at very different sizes depending on distance or camera quality. A tree far in the background of one picture and the same tree up close in another still look like the same object. People recognize this easily, but computer features must be explicitly designed to handle changing resolution. Multi-scale feature pyramids or representations that use relative rather than absolute measurements help preserve similarity across sizes.
  - **Resolution invariance** fits naturally into the broader goal of perceptual invariance and extends the idea of scale stability. When resolution changes, pixel-level detail can shift a lot while the basic structure of the scene usually stays the same. A high-resolution image shows fine textures, such as fabric patterns or wood grain, while a low-resolution version preserves only broad shapes and color regions. People still recognize the content easily, so feature systems should do the same.
  - **Lighting invariance** is essential because illumination often changes in unpredictable ways. A face lit from the side, under soft, diffuse light, or in harsh midday sun is still the same face. Features that depend on absolute brightness will fail in those situations. Methods that focus on edges, gradients, or reflectance rather than raw intensity are more robust to lighting changes.
  - Finally, a perceptual feature system **must tolerate noise and imperfections**. Real images often have sensor noise, compression artifacts, banding, or color quantization, especially in low-quality or heavily compressed photos. A system that treats every small distortion as meaningful cannot judge similarity reliably. Smoothing, robust statistical measures, and representations that capture global structure instead of local pixel irregularities help reduce sensitivity to these artifacts.

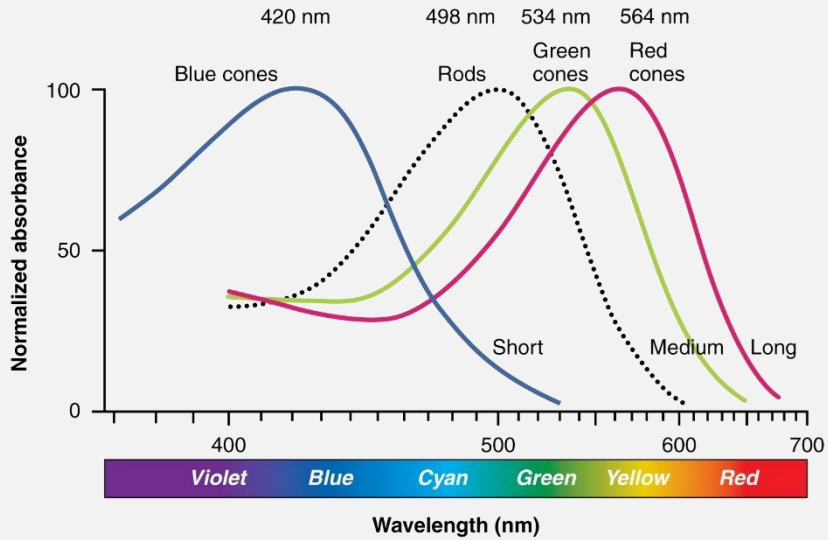
- Physical basis of visual perception:
  - Every image comes from how illumination interacts with surface reflectance. **Illumination** $l(x, y, z)$, is the amount of light at a point in space, measured in lux and weighted by the eye's sensitivity to different wavelengths. **Reflectance** $r(x, y, z)$, is the fraction of that light a surface reflects back toward the viewer.
  - The upper image on the right shows this interaction on a single surface fragment. Light from the source hits the material, and the material reflects different wavelengths in different directions. The viewer sees only the light that reaches their line of sight. Small changes in lighting direction, intensity, or spectral composition can change how the same surface looks, because what we see is a combination of illumination and reflectance.
  - The second image shows typical reflectance curves for natural materials. Snow reflects light across most wavelengths, vegetation shows strong peaks in the green and near-infrared regions, and water reflects very little. These curves show that the physical signal is rich and continuous, while human vision and camera sensors reduce it to just a few channels. Apparent color therefore depends as much on lighting as on the material's properties.
  - Typical illuminance and reflectance values show this variability. Sunlight can be tens of thousands of lux, indoor lighting a few hundred. Bright surfaces reflect most of that light, while dark surfaces reflect only a small part. This explains why pixel values do not directly match perceived color or brightness. Perceptual features must account for how people interpret the combined effects of lighting and material, not just the raw physical signal.



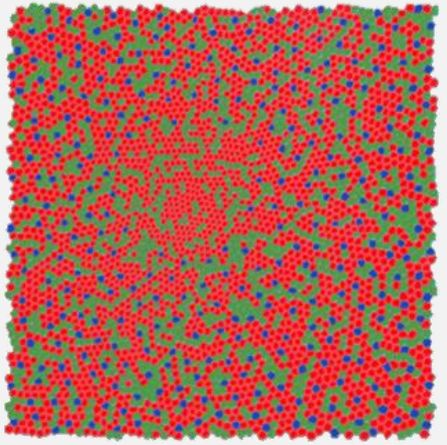


- The eye captures light and converts different wavelengths into electrochemical signals the brain can interpret. Light enters through the cornea, passes the pupil, and then reaches the lens. These parts form an adjustable optical system that focuses images of objects at different distances. The system also shifts for different light levels, similar to a camera's aperture and focus. The lens projects an inverted image onto the retina at the back of the eye.
  - The retina has photoreceptors called rods and cones, each serving a different role in vision. Cones are concentrated in the central retina, especially in the macula and its center, the fovea, and they provide sharp, detailed, and color vision. There are three types of cones, each sensitive to a different range of wavelengths.
    - L cones detect long wavelengths and peak near 564 nanometers, which corresponds to red
    - M cones respond to medium wavelengths and peak near 534 nanometers, associated with green
    - S cones are sensitive to short wavelengths and peak around 420 nanometers, enabling blue perception
  - Rods are more numerous in the retina's peripheral regions and work best in low light. They are essential for night vision and peripheral awareness but cannot detect color. Rods and cones convert incoming photons into neural signals that the brain turns into images.





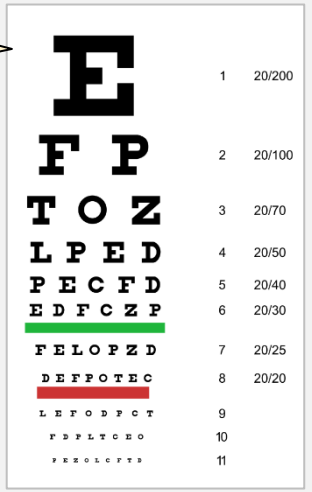
- The human eye has about 6 million cones and 120 million rods. Cones come in three types:
  - Roughly 1 percent S-cones that are sensitive to blue light,
  - 39 percent M-cones that detect green, and
  - 60 percent L-cones that detect red.
- Near the center of vision around the fovea, the share of blue-sensitive cones can rise to as much as 7 percent. These proportions vary a lot between people, which affects color perception and can cause color blindness when some cones are missing or do not work.
- Visual acuity measures how sharp and clear vision is. It reflects the eye's ability to see fine details. The standard test uses a Snellen chart. On that chart, 20/20 means the eye can clearly identify objects about 1.75 millimeters apart from 20 feet, which is a visual angle of roughly one arcminute (1/60 of a degree). By contrast, 20/40 means the person must be at 20 feet to see what someone with normal vision can see at 40 feet, showing reduced sharpness.

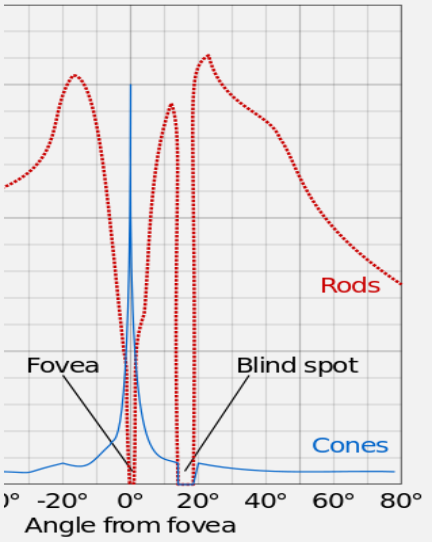


Ratio	Metric	Snellen	Arcminutes
2,0	6/3	20/10	0.5′
1,33	6/4,5	20/15	0.75′
1,0	6/6	20/20	1′
0,8	6/7,5	20/25	1.25′
0,67	6/9	20/30	1.5′
0,5	6/12	20/40	2′
0,4	6/15	20/50	2.5′
0,2	6/30	20/100	5′
0,1	6/60	20/200	10′
0,05	6/120	20/400	20′

1.4′ or less is required to drive a car

Standard Snellen Chart

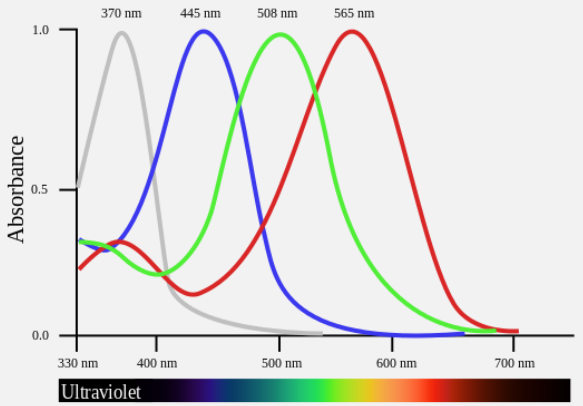




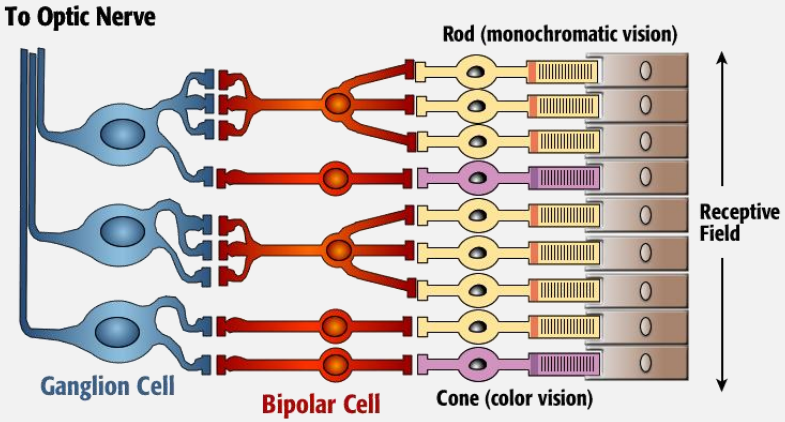
- When comparing visual abilities across species, animals perceive the world very differently. For example, a cat's visual acuity is about 20/100, much lower than a human's. Cats have only two cone types, sensitive to blue (around 450 nm) and yellow (around 550 nm), so their color vision is limited. They see 6-8 times better than humans in low light because they have many rods and a reflective layer behind the retina called the tapetum lucidum. Their field of view is about 200 degrees versus the human average of 180 degrees, which gives them broader peripheral vision for hunting but a less detailed, somewhat blurred image compared to human vision.

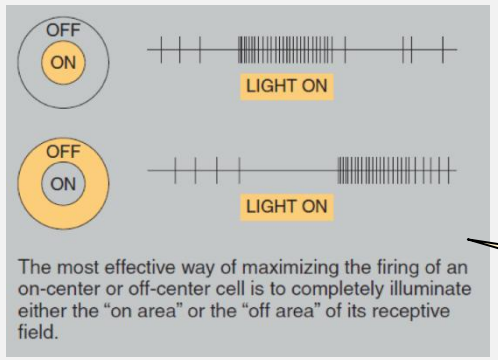
- Dogs also have dichromatic vision with blue and yellow cones. Their visual acuity is about 20/75, a bit better than cats but still less sharp than human vision. Larger animals such as elephants have lower acuity, around 20/200. Small rodents see much less clearly, about 20/800. Insects like bees have acuity near 20/1200. Flies have the poorest acuity, roughly 20/10800.
- On the other side, birds of prey such as eagles have extraordinary vision that is about 4-5 times sharper than average human vision. This clarity lets them spot prey from long distances. Many bird species are tetrachromatic; they have four types of cones and can see a wider range of colors, including ultraviolet light that humans cannot see. Similarly, goldfish and zebrafish have four distinct cone types. The extra cone is usually sensitive to ultraviolet wavelengths that peak near 370 nm, helping them detect UV light underwater.

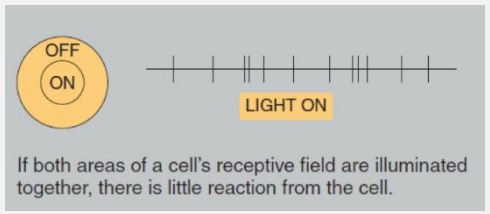
Goldfish and zebrafish are tetrachromatic



- Processing of visual information already starts in the retina, an approach that inspired the use of convolutions in deep learning. Rods and cones show a distinctive chemical behavior: they release glutamate in the dark and stop releasing it when exposed to light. This is unusual for a sensory system but releasing neurotransmitters in the dark helps to support regeneration during sleep.
  - Bipolar cells connect to either rods or cones, never both. On-bipolar cells become active and fire in light, while off-bipolar cells stop firing in bright conditions.
  - Next, ganglion cells form the first receptive fields by combining signals from several bipolar cells. These ganglion cells act like edge detectors, comparing the light level in a central region with that in the surrounding area. On-center ganglion cells fire when the center is bright and the surround is dark. Off-center ganglion cells fire when the center is dark and the surround is bright.
- Additional retinal cell types, such as horizontal cells and amacrine cells, act as inhibitors to sharpen contrast. However, this enhanced contrast can sometimes cause dark and light boundaries to appear exaggerated, either under- or oversaturated. This effect, known as lateral inhibition, provides negative feedback to neighboring cells to amplify the difference between strong and weak signals. While this improves contrast perception, it can also create after-images, where visual impressions persist even after the original stimulus is gone.

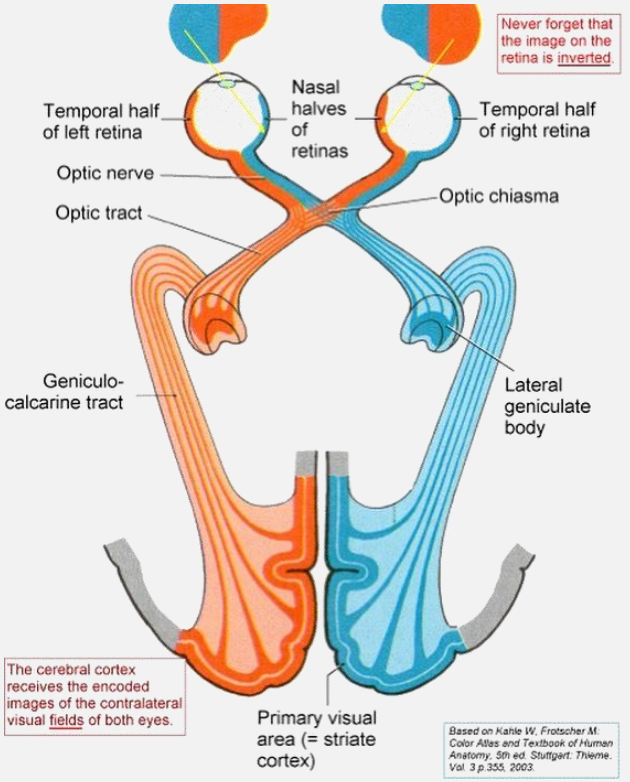


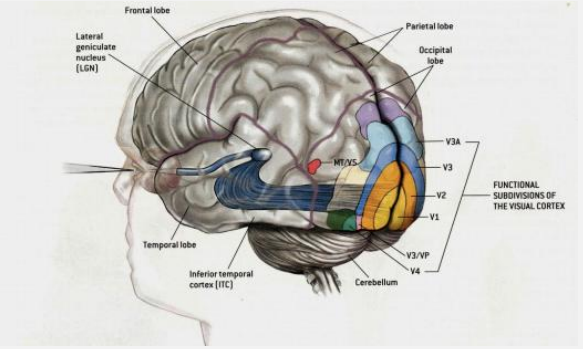




Different Ganglion Cells at work for their receptive field

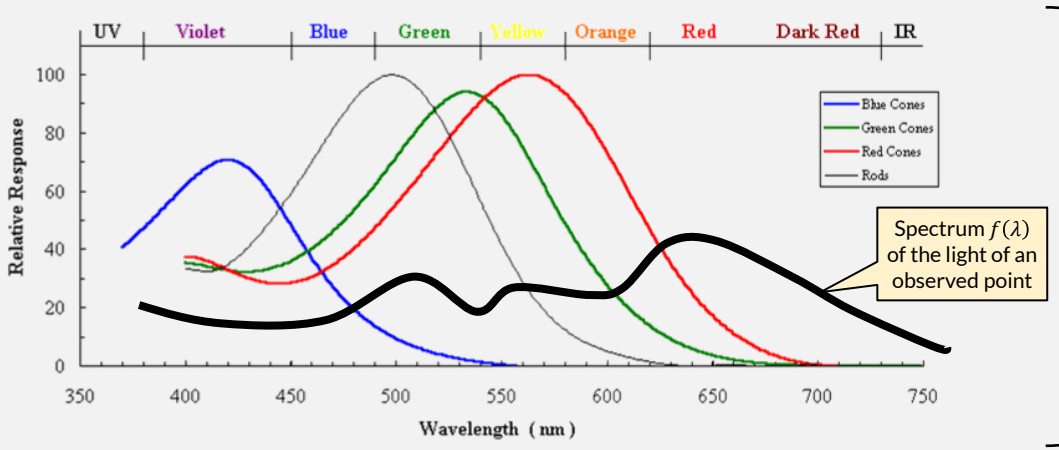
- The **Lateral Geniculate Nucleus (LGN)** performs receptive field functions similar to retinal ganglion cells and receives strong feedback from the visual cortex. It separates the two visual fields: the left visual field is processed by the right side of the brain and the right visual field by the left. The LGN also combines input from both eyes. Its first two layers emphasize rod-based signals for movement and contrast, while the next four layers emphasize cone-based signals for color and finer details of form.
- The **Primary Visual Cortex (V1)**, detects edges and orientations. Some V1 neurons respond only when a pattern appears in a specific position, while others respond to the same pattern regardless of position. Neurons in the visual cortex fire when particular patterns appear in their receptive fields. Lower levels of the system process simpler patterns. Higher levels handle more complex patterns, such as face detection. The information stream then follows two paths to those higher levels.
  - The **Ventral Stream** handles form recognition and object representation and is connected to long-term memory.
  - The **Dorsal Stream** processes motion and object location and also helps coordinate eye and arm movements, such as reaching for an object.
- **Cortical magnification** means the brain uses more neurons to process signals from the center of the visual field, especially the fovea. Although the fovea is small on the retina, it takes up a much larger area in the visual cortex. This is because the fovea has a high density of cones, which provide detailed, color-rich vision. The brain magnifies that input so we can see fine detail, such as reading or recognizing faces.
- The human visual system and deep learning models for image classification share similarities and differences. Both detect features, process information in layers, and learn from examples. The human system is biological, has limits on processing speed and scalability, and adapts well to varied visual input. Deep learning models are artificial, can process very large amounts of data quickly, and become highly efficient at specific classification tasks.





11.2 Color Information

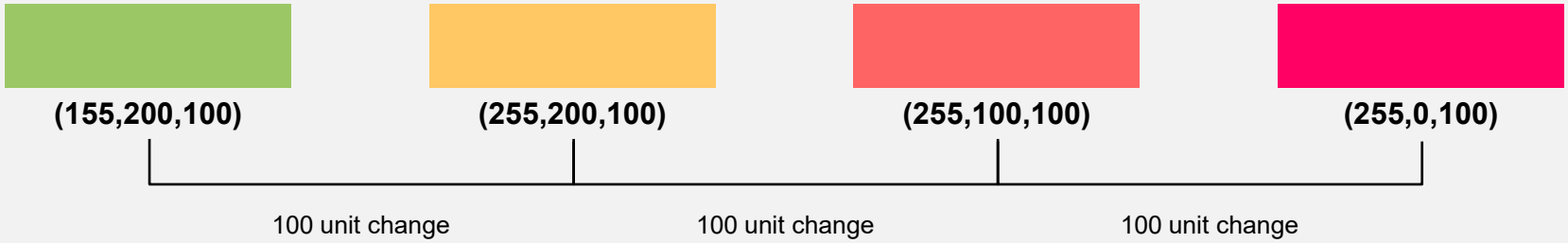
- Color perception is the eye's way of representing how energy is spread across electromagnetic wavelengths. It is only an approximation because reducing that distribution to three values loses much information, so two different spectra can look identical.
- This approximation lets us reproduce human color perception using three additive components that emit wavelengths matching the sensitivity of the red, green, and blue cones. These components form the RGB family, which is designed for human vision and may not work for animals with different cone sensitivities. Besides RGB, many other color systems exist. The most common systems include:
  - **CIE** – Established by the International Commission on Illumination (CIE) to establish a link between the physical signal and the perception of a typical human observer.
  - **RGB** – The prevailing system since the introduction of sRGB by HP and Microsoft in 1996.
  - **HSL/HSV** – This converts Cartesian RGB coordinates to cylindrical coordinates for hue and saturation, while incorporating luminance/brightness as the third component.
  - **YUV** – Utilized in NTSC and PAL signals and serves as the foundation for numerous image and compression algorithms like JPEG and MPEG (using YCbCr), which are not discussed further.
  - **CMYK** – In printing, this method subtracts color from an initially white canvas. The ink absorbs light, and a blend of various inks generates the intended color, which is not covered in further detail.



When we observe the emitted or reflected light spectrum $f(\lambda)$, we obtain 3 (or 4) values for each cone type (and rod). To calculate intensity, we multiply the observed spectrum by the sensitivity filter of the cones (e.g., $c_{red}(\lambda)$), then integrate the result across all wavelengths. For example, for the red cone:

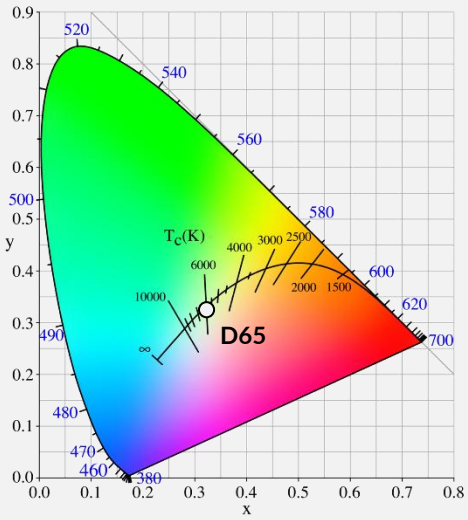
$$red = \int_0^\infty f(\lambda) \cdot c_{red}(\lambda) d\lambda$$

- To extract perceptual color features and use distance measures such as Euclidean distance, we need a color representation that matches human perception. Consider the four colors in sRGB space shown below. Between adjacent boxes, only one channel changes and the color distance is 100 units. Even so, we perceive these shifts differently. The jump from green to yellow (first to second) looks large, while the jump from orange to red (third to fourth) looks much smaller. This happens because sRGB maps light nonlinearly.



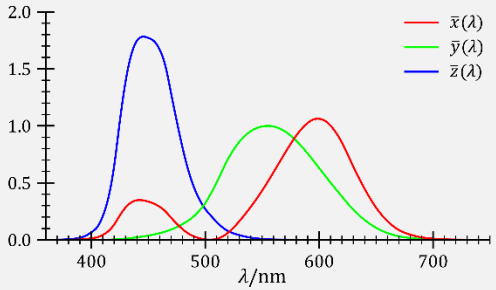
- Human vision uses three types of cones, so three components are needed to represent the full range of colors. Color can be divided into:
  - **Brightness** is the visual perception of emitted or reflected light. It depends on an object's luminance but is not directly proportional to it. Instead, the observer interprets it subjectively.
  - **Chromaticity** is the objective description of color without luminance. It has two independent components: hue and saturation. Chromaticity diagrams show the visible color range, called the color gamut (see right).
    - **Hue** indicates how closely a color matches pure colors like red, green, or blue. Hue values lie along the edge of the chromaticity diagram and are measured in degrees around the white point, for example D65.
    - **Saturation / Chroma / Colorfulness** quantifies how much light is spread across the visual spectrum. Pure or highly saturated colors are concentrated near a single wavelength with high intensity. In the chromaticity diagram, saturation is the distance from the white point relative to the maximum possible distance in that direction. Note that green lies much farther from white than red or blue.

source:
https://en.wikipedia.org/wiki/Color_temperature



11.2.1 Color Spaces

- The CIE defined color spaces to faithfully represent how people see color. The mathematical relationships behind these spaces are important for advanced color management. In 1931 the CIE introduced the **XYZ color space** to model human color perception. Their experiments showed that people see green as brighter than red or blue when all have the same physical intensity. At night, rods take over from cones, producing mostly monochrome vision with finer sensitivity to brightness changes.
  - The definitions of X, Y, and Z differ from the usual additive or subtractive primary color model. Y denotes luminance. X and Z indicate chromaticity without brightness. Y corresponds to M-cone sensitivity (green), Z to S-cone sensitivity (blue), and X is a combination of cone responses.
  - To calculate X, Y, and Z from spectral data, a standard colorimetric observer was defined by experiments. This observer represents the average chromatic response of a typical person within a 2 degree field in the fovea, the central region where most cones are concentrated. The color matching functions $\bar{x}(\lambda), \bar{y}(\lambda)$ and $\bar{z}(\lambda)$ define the spectral weights for the measured spectral radiance or reflectance $f(\lambda)$. We compute X, Y, and Z by integrating over wavelengths from 380 nm to 780 nm:



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

$$Z = \int_{380}^{780} f(\lambda) \cdot \bar{z}(\lambda) d\lambda$$

source: https://en.wikipedia.org/wiki/CIE_1931_color_space

  - One key strength of CIE XYZ is its completeness. It covers every color visible to the human eye, making it useful for specifying and analyzing color. However, many colors in the CIE XYZ space fall outside the gamut of standard RGB spaces, because RGB systems cannot reproduce the entire visible spectrum.
  - CIE XYZ, widely used in colorimetry, color management, and image processing, forms the basis of color spaces such as CIE Lab, CIE Luv, and CIE LCH. These spaces are used for color communication and calibration in printing, photography, and display technology.

- The **CIE xyY space**, introduced in 1931, was the first attempt to separate chromaticity from luminance. In the CIE XYZ system, the Y value was designed to represent the standard observer's perceived luminance. The x, y, and z components are obtained by normalizing the XYZ values.

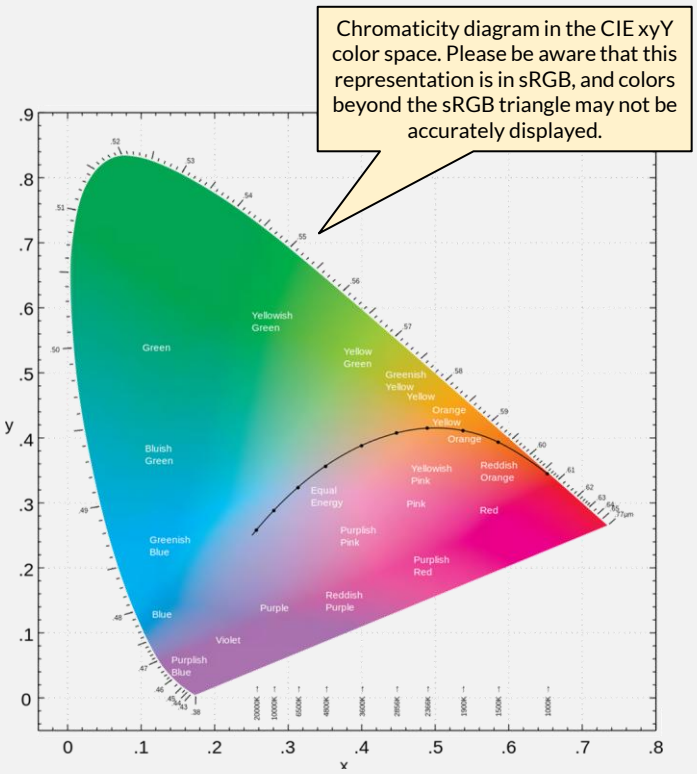
$$x = \frac{X}{X+Y+Z} \qquad y = \frac{Y}{X+Y+Z} \qquad z = \frac{Z}{X+Y+Z} = 1 - x - y$$

  - The resulting color space comprises x, y, and Y. The x and y values define the chromaticity diagram, shown in the lower right section of the page, and represent color without luminance. CIE xyY is a widely used color specification that covers all visible colors for the standard observer. Note that the chromaticity diagrams here are shown in the sRGB space, so they do not display the full CIE xyY color gamut. To reverse the transformation from x, y, and Y values, follow these steps:

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

  - The outer curve of the chromaticity diagram, called the spectral locus, corresponds to wavelengths measured in nanometers. The figure shows those wavelengths in micrometers. The CIE xyY space describes color as seen by the standard observer. It does not represent an object's inherent color because perceived color changes with the lighting and the light source's color temperature. In low light the human eye cannot see color and is limited to shades of gray.
  - This color space is versatile and essential for accurate, consistent color across different devices and applications. By including all colors seen by the standard observer, it links human perception to digital color systems and ensures precise, uniform color reproduction and interpretation.

source: https://en.wikipedia.org/wiki/CIE_1931_color_space



- The CIE xyY space covers the entire human visible color range, but it is not perceptually uniform. Two colors the same distance apart can look very different depending on where they are in the space. The **CIE Lab color space** (1976) was created to provide a more perceptually uniform scale. It also exceeds the gamut of many other color spaces and is device independent, so it is commonly used to map colors between different systems.
  - The L component represents lightness and is based on luminance Y, adjusted for perception to produce a uniform scale so that a one-unit change corresponds to an equal perceived change in lightness. L typically ranges from 0 to 100, with $L = 0$ corresponding to black and $L = 100$ corresponding to white.
  - The component a represents the red-green opponent axis. Negative values indicate green and positive values indicate red. These values usually range from -128 to 127. When the component equals 0, it represents a neutral gray.
  - The b component shows the blue/yellow opponent axis. Negative values mean blue and positive values mean yellow. Values normally range from -128 to 127. When $b = 0$, the color is neutral gray.

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

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

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

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

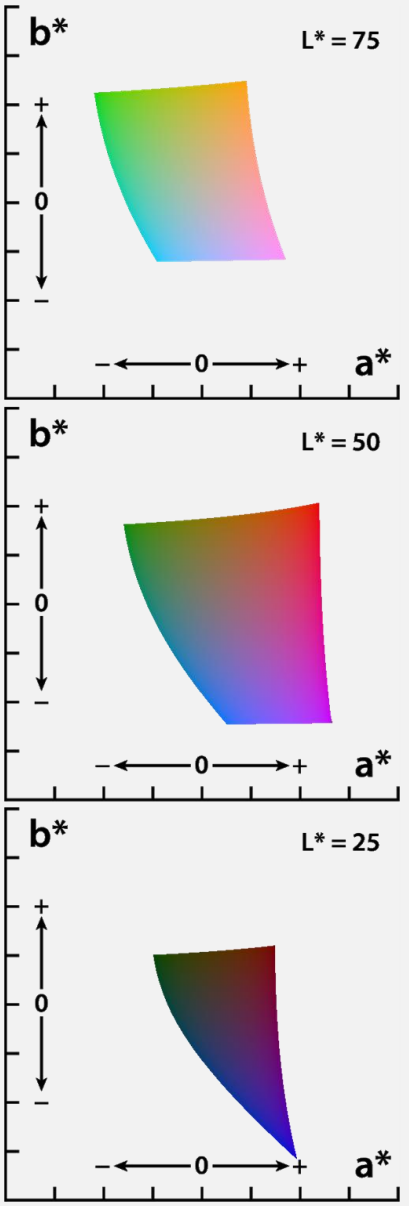
$$f(t) = \begin{cases} \sqrt[3]{t} & \text{if } t > \left(\frac{6}{29}\right)^3 \\ \frac{841 \cdot t}{108} + \frac{4}{29} & \text{otherwise} \end{cases}$$

$$X_n = 242.364495$$

$$Y_n = 255.0$$

$$Z_n = 277.67358$$

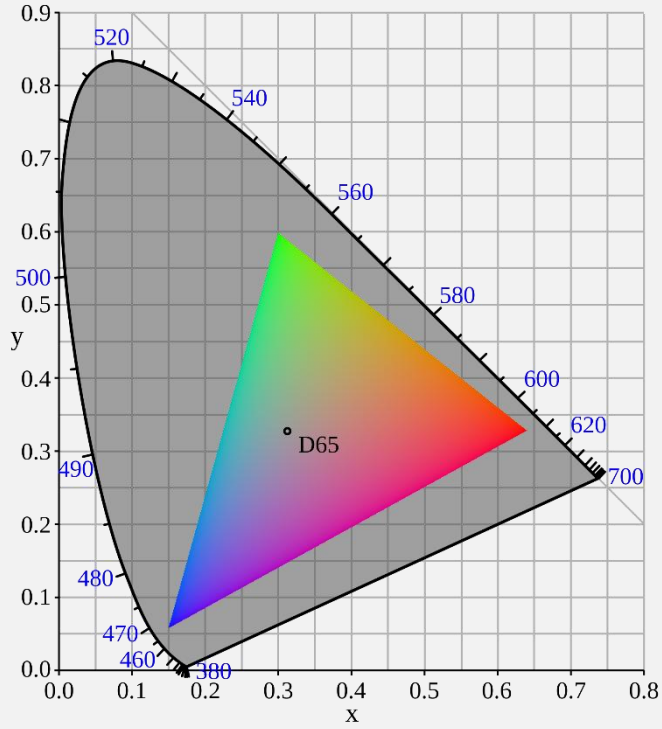
source: https://en.wikipedia.org/wiki/CIELAB_color_space



- RGB is the main color model used in computing. HP and Microsoft introduced **sRGB** as an additive model for monitors, printers, and the Internet. It is formalized as IEC 61966-2-1:1999 and is the default when no color model is specified.
  - sRGB uses the ITU-R BT.709 primaries (also called Rec. 709) to define its color gamut (the range of colors it can produce). It became widely used because it matched the standard CRT monitors of its 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

source: https://en.wikipedia.org/wiki/SRGB



  - In sRGB, colors lie inside the triangle in the figure on the right when only nonnegative values are used. This color gamut does not cover all chromaticities and misses a large part of the green and blue range.
  - The sRGB scales are nonlinear, with an approximate gamma of 2.2. To convert from linear RGB to sRGB, the specification gives functions that map each channel value. Let c_{sRGB} represent a channel value (red, green, or blue) in sRGB, and c_{linear} denote the corresponding value in linear RGB. Both range from 0 to 1. For quantized values, divide by or multiply by $(2^{\text{bits}} - 1)$.

$$c_{sRGB} = \begin{cases} 12.92 \cdot c_{linear} & \text{if } c_{linear} \leq 0.0031308 \\ 1.055 \cdot c_{linear}^{\frac{1}{2.4}} - 0.05 & \text{otherwise} \end{cases}$$

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

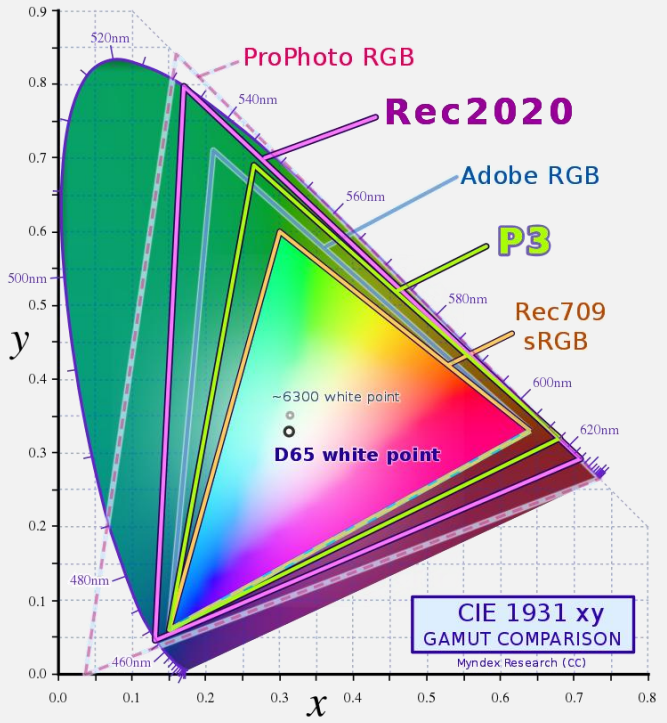
- The conversion from CIE XYZ to linear RGB is as follows:

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

  - Note that the transformation above maps linear RGB to XYZ. To obtain sRGB values, apply an additional transformation as described on the previous page.
  - Note that the RGB space does not cover the entire XYZ space, which represents the colors humans can see. If a mapping produces values outside the 0 to 1 range, set them to the nearest limit: 0 for negative values and 1 for values greater than or equal to 1.

- RGB values are often quantized into integer ranges. This uses simple multiplication and division by $2^{\text{bits}} - 1$. For example, true color normally uses 8 bits per channel, so the multiplier is 255. Deep color uses 16 bits per channel, so the multiplier is 65535. In some cases, quantization is based on a set of reference colors, or a color palette. Then a color is represented by its nearest match in the palette.
- Apart from sRGB and linear RGB, several other RGB models have been introduced. You can create an RGB space simply by choosing the primaries and the white point. These alternative models extend the original sRGB, which covers a limited color gamut, to include a wider range of colors. For example, Rec. 2020 is designed for ultra high definition television (UHDTV) and has a much wider gamut than HDTV based on Rec. 709. Some RGB models even go beyond the standard chromaticity chart to include more of the green and blue regions.

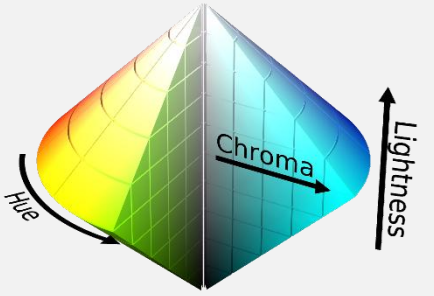
source: https://en.wikipedia.org/wiki/Rec._2020

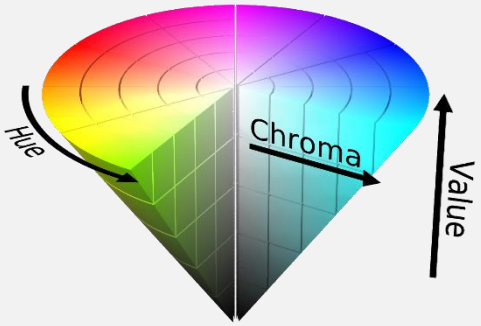


- Artists typically begin with a relatively bright color and then apply
  - white to "tint" the color,
  - black to "shade" the color, or
  - both white and black (gray) to "tone" the color.

  To support these techniques in computer graphics, the HSL and HSV color models provide alternate representations of RGB that make color manipulation easier. Both models use hue and chroma to define chromaticity. HSL adds lightness and places fully saturated colors at $L = 1/2$. It allows tinting ($L = 1$) and shading ($L = 0$) without changing saturation. HSV uses value and places fully saturated colors at $V = 1$. It allows shading ($V = 0$) without changing saturation, while tinting does change saturation.

source: https://en.wikipedia.org/wiki/HSL_and_HSV





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

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

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

$$C = M - m$$

$$H = 60° \cdot H'$$

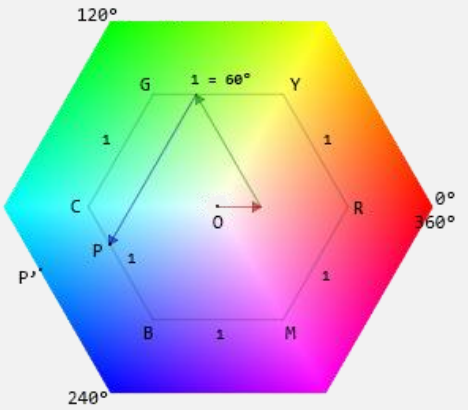
$$V = M$$

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

$$H = 60° \cdot H'$$

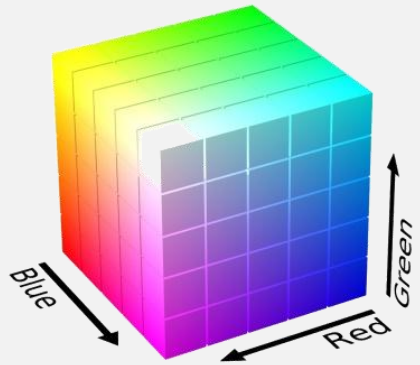
$$L = \frac{1}{2}(M + m)$$

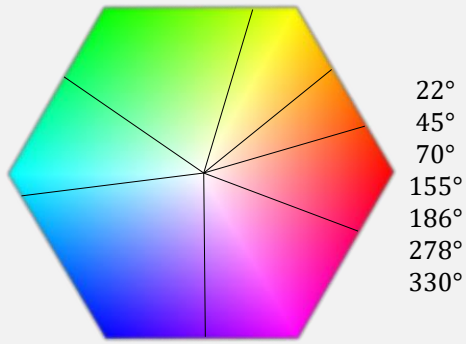
$$S_{HSL} = \begin{cases} 0 & \text{if } L = 1 \\ \dfrac{C}{1 - |2L - 1|} & \text{otherwise} \end{cases}$$



11.2.2 Color Features

- **Color Histograms:** Histograms are a simple way to show a color distribution using a fixed set of reference colors that serve as the dataset's vocabulary. Each pixel is assigned to the nearest reference color, and the counts of each reference color in the image are tallied. To make the result scale invariant, divide the counts by the total number of pixels; the normalized values then give the probability of each reference color appearing.
  - Selection of reference colors:
    - The simplest method is to quantize the R, G, B values in linear RGB space, as shown on the right. For example, using 2 bits gives 4 uniform levels per channel, producing $4^3 = 64$ reference colors, often written c_i with $1 \leq i \leq 64$. You can pick any number of uniform ranges, for example 5, to get the number of reference colors you want.
    - For better perceptual color matching, a nonuniform distribution is preferred. In HSV space, the color hexagon can be split into regions of perceptually similar colors, as shown on the right. To reflect greater sensitivity to brightness, use more bins along the V dimension. With 7 chromaticity values and 9 V bins, we obtain 63 reference colors, denoted as c_i.
    - For uniform color spaces like CIE Lab, use uniform ranges. Give the L axis more ranges than the a and b axes to account for differences in brightness sensitivity.
    - We quantify the similarity between two reference colors c_i and c_j by measuring the distance between their representations, for example $d_{i,j}$. In Cartesian coordinates this is the Euclidean distance between the centers that represent the colors. In cylindrical coordinates, as in HSV, compute the angle difference as $\min(|\alpha - \beta|, 2\pi - |\alpha - \beta|)$ and combine it with the other channel differences using a Manhattan distance. In all cases normalize each channel to a common range, for example $[0,1]$, before computing distances.





- Comparison of histogram (distance measure)
  - Let h_i and g_i be the normalized histograms of two images, arranged by N reference colors c_i with $0 \leq h_i, g_i \leq 1$. Although color quantization uses a three dimensional color space, the histograms are one dimensional because they count the reference colors. We also compute distances $d_{i,j} = d_{j,i}$ between pairs of reference colors c_i and c_j.
  - An initial simple approach is to calculate the Manhattan or Euclidean distance between histograms:

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

    These distance formulas work effectively, but they do not consider the similarity between reference colors. A slight shift in lighting or color representation can result in significant distance variations.
  - To address cross-correlation between reference colors, we must apply a quadratic distance measure and utilize a matrix $\mathbf{A}$ based on the distances between reference colors:

$$\delta_{quadratic}(\boldsymbol{h}, \boldsymbol{g}) = (\boldsymbol{h} - \boldsymbol{g})^{\top} \mathbf{A} (\boldsymbol{h} - \boldsymbol{g}) \qquad \mathbf{A}\text{: } a_{i,j} = 1 - \frac{d_{i,j}}{\max\limits_{k,l} d_{k,l}}$$

Distance normalized by maximum distance for all pairs of reference colors

  - When a user submits a sketch as a query or specifies the desired colors in an image, histogram intersections (equivalent to a partial match query) are more suitable. Let $g_i \neq 0$ represent the user-selected colors, and $g_i = 0$ denote 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 method uses luminance or brightness histograms and ignores color. First compute brightness, for example the L^* channel from CIE Lab. Then divide the luminance values into several uniform ranges. The next steps follow the methods described above, including computing quadratic distances to measure similarity between brightness values. The resulting features describe the image's brightness and are often used for shot detection in videos because changes in lighting often mark shot boundaries.
  - We can quantize chromaticity only and ignore brightness or luminance. Suitable color spaces for this are CIE Lab, CIE LCH, HSL, and HSV. These features describe how color is distributed and are robust to changes in lighting, as long as lighting does not greatly change perceived chromaticity. They also help find dominant colors, so users can search for blue horses by chroma instead of specifying an exact RGB value.
- **Discussion**:
  - Histograms are simple but usually produce good results. They are robust to translation, rotation, noise, and scaling, and in some cases to changes in lighting.
  - Ignoring spatial relationships between pixels can produce unexpected results. For example, the blue lake at the bottom of the picture might match the blue sky at the top and a blue car in the middle. Because color similarity does not consider pixel locations, it is easy to create two images with the same histogram but different content.
  - The histogram intersection method helps a retrieval system find the color of the main objects. Users select a color, and the search is extended with a histogram subquery that uses the intersection technique.
  - Color histograms often have very high dimensionality. 64 dimensions may be a minimum for effective retrieval, and they can exceed a 1000. Searching in such high dimensional spaces uses a lot of resources and is inefficient. Dimensionality reduction methods, such as principal component analysis (PCA), can reduce the number of dimensions and address correlations among reference colors.
  - Best results come from quadratic distance functions, which account for similarities between reference colors. To avoid costly vector-matrix-vector multiplications during search, we can use the eigenvectors of matrix A to transform histograms into a space where Euclidean distance applies.

- **Color Moments:** Statistical moments offer another way to describe color distributions in the chosen color space. Any of the earlier color spaces can be used, but perceptually uniform spaces such as CIE Lab are preferred for measuring distances and similarities. We typically use Lab rather than LCH to avoid dealing with angular differences.
  - Single-channel moments calculate statistical values for each channel (L, a, b). If c represents a color channel, N and M denotes the number of rows and columns in the image, the first four moments are defined as follows:

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

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

    Mean μ_c and variance v_c indicate the position and width of the distribution peak. Skewness s_c reveals if the peak is skewed left or right. Kurtosis k_c identifies the presence of outliers (values far from the mean). When considering three channels, this method yields 12 feature values describing the distribution in each channel.
  - We can calculate additional covariance values between pairs of channels. If we have three channels, we get three extra covariance values from the possible channel pairs:

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

  - We now can define a color moment as follows: Using the CIE Lab color space, we extract 12 moments and 3 covariances, for a total of 15 values. We arrange these values into a vector $\boldsymbol{m}$ (in a fixed order) and compare it to the feature vectors $\boldsymbol{m}_i$ and $\boldsymbol{m}_j$ from two images using either Euclidean or Manhattan distance.

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

- When calculating moments, the formulas can be rewritten so one pass computes all values (where c represents a color channel):

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

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

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

$$k_c = \frac{a_{c,4} - 4a_{c,3} \cdot a_{c,1} + 6a_{c,2} \cdot a_{c,1}^2 - 3a_{c,1}^4}{v_c^2} \qquad cov_{c_i,c_j} = b_{i,j} - \mu_{c_i} \cdot \mu_{c_j}$$

- **Variants**:
  - Like histograms, we can compute moments for brightness (luminance) only. This removes the need for covariance and gives four brightness moments. Alternatively, we can compute moments for chromaticity only, ignoring brightness. That case yields eight moments plus one covariance value, forming a nine-dimensional feature.
- **Discussion**:
  - The value ranges of moments can vary significantly. To apply a distance measure, we must first normalize the values into the same range. Scaling can be achieved by dividing with the standard deviation of the values along that dimension as outlined in the chapter on Vector Search.
  - Color moments, like histograms, are robust to translation, rotation, noise, and scaling. They can also tolerate some changes in lighting. However, like histograms, they ignore spatial relationships between pixels, which can lead to unexpected results.
  - Color moments are independent and do not need a cross-correlation matrix to compute quadratic distances. Their vectors are much shorter, typically about 15 values when all moments are used, while histograms can have up to 1000 bins. This compact representation speeds up searches without reducing retrieval quality.

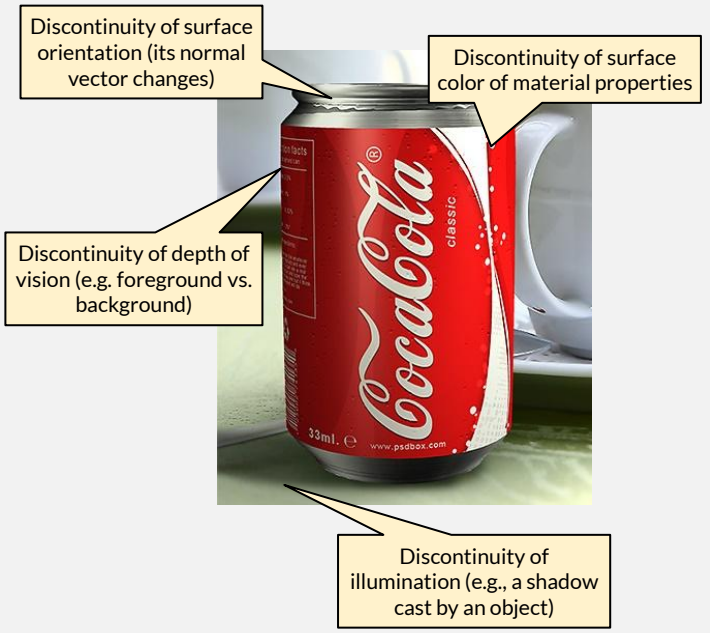
11.3 Texture Information

- Texture is key to image retrieval because it gives information about how colors, patterns, and shapes are arranged in an image. Color mainly refers to hue and saturation, while texture describes how those colors are distributed and organized. Texture also reveals surface details like roughness, smoothness, or grain, which help recognize and distinguish objects and scenes.
- In image retrieval, systems use texture features together with color features to improve accuracy and robustness. Analyzing both color and texture gives more complete and reliable results. This combination helps systems better understand image content and makes it easier to find relevant images in large databases.
- Texture describes the structure of a surface or a part of an image. It shows how color is spread in space, how that spread changes, and the direction and frequency of those changes. Texture analysis can be done in three ways:
  - **Structural approach:** Identify groups of basic elements called "texels" that form regular, repeating patterns, as shown in the examples below. Color information is usually ignored.

    When the focus is specifically on identifying recurring spatial patterns or textures in satellite imagery to distinguish categories such as crops, water, or forest, it is often described as texture-based land cover classification or remote sensing image classification.
  - **Statistical approach:** Analyze pixel neighborhoods, measure how pixels are arranged, and produce statistical summaries like histograms and moments. We use edge detection and filters to extract texture features.
  - **Fourier approach:** Apply the Fourier transform to convert the image into the frequency domain, then extract information that indicates the presence of Gabor filters in that domain.
- Texture analysis is often done on grayscale images. In those cases, we can use the Y or L component from the CIE color models. The original image must be converted from sRGB to linear RGB before converting to CIE XYZ or CIE Lab, as described in sRGB to linear RGB conversion. For this discussion, we focus on monochrome images with only a brightness or luminance channel. More advanced methods may also use chromaticity information for texture analysis.

- **Edge magnitude and direction** (structural approach)
  - Edges in images result from various factors, as shown on the right. Detecting edges means finding large gradients or sudden changes between neighboring pixels, which mark boundaries between image regions. A common method is to smooth the image with a Gaussian and then convolve it with the Sobel operator. This produces the gradient values in the x and y directions, called g_x and g_y. The kernel matrices used are shown below:



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

A kernel is applied to an image by sliding it one pixel at a time and computing the weighted sum of the pixels it covers at each position. For the Sobel operator, the horizontal kernel highlights intensity changes from left to right and produces g_x, while the vertical kernel highlights changes from top to bottom and produces g_x.
At each pixel, the kernel multiplies its weights by the local pixel intensities, sums the products, and writes that value to the output image. Repeating this over the whole image produces the full g_x and g_y gradient maps. We can then calculate the gradient magnitude, denoted as $g_{\text{mag}}(x, y)$, and the gradient direction, represented as $g_{dir}(x, y)$, using the following formulas:

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

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

  - After the transformation above, we get two values for each image pixel. The first value gives the change magnitude (energy). The second value gives the change direction, from darker to lighter. A value of zero corresponds to a vertical edge, with the change direction perpendicular to the edge and the lighter pixel on the right.

- We can now generate straightforward texture-based features.
  - **Edge Density of Image**: The percentage of the image with gradient magnitude $g_{mag}(x,y) \geq \tau$ for a specified threshold τ. This indicates the presence of edges with sufficient energy. Continuous regions in the image, like the sky or a lake, yield lower values, while images with multiple objects or cityscapes have higher values.

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

  - **Gradient Histograms:** This method is analogous to the color histogram approach, but we now have only two values per pixel to quantify (direction and magnitude). Remember that differences in direction are computed as $\min(|\alpha - \beta|, 2\pi - |\alpha - \beta|)$. We must normalize energy and direction ranges to calculate the distance $d_{i,j}$ between two reference gradients. This allows us to compute the matrix $\mathbf{A}$ for the quadratic distance measure. When given two histograms $\boldsymbol{h}$ and $\boldsymbol{g}$, and assuming N reference gradients, we obtain distances as follows:

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

$$\delta_{quadratic}(\boldsymbol{h},\boldsymbol{g}) = (\boldsymbol{h}-\boldsymbol{g})^{\top}\mathbf{A}(\boldsymbol{h}-\boldsymbol{g}) \qquad \mathbf{A}\text{: } a_{i,j} = 1 - \frac{d_{i,j}}{\max\limits_{k,l} d_{k,l}}$$

Distance normalized by maximum distance for all pairs of reference gradients

  - **Gradient Moments:** We compute moments for both magnitude and direction and compute a covariance between them. Let X denote either magnitude or direction. This produces nine feature values that describe the gradient distribution:

$$\mu_c = \frac{1}{N \cdot M} \sum_{x,y} g_c(x,y) \qquad v_c = \frac{1}{N \cdot M} \sum_{x,y} (g_c(x,y) - \mu_c)^2 \qquad s_c = \frac{1}{N \cdot M} \sum_{x,y} \left(\frac{g_c(x,y) - \mu_c}{\sqrt{v_c}} \right)^3$$

$$k_c = \frac{1}{N \cdot M} \sum_{x,y} \left(\frac{g_c(x,y) - \mu_c}{\sqrt{v_c}} \right)^4 \qquad cov_{mag,dir} = \frac{1}{N \cdot M} \sum_{x,y} \left(g_{mag}(x,y) - \mu_{mag}\right) - \left(g_{dir}(x,y) - \mu_{dir}\right)$$

- **Laws' Texture Energy** (structural approach)
  - Laws texture masks calculate 9 values for each pixel in the image to capture different texture features. These masks are derived from 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 generate 16 matrices by multiplying pairs of prototype vectors. For example, for E5L5, we obtain the kernel matrix $\mathbf{G}_{E5L5}$ as follows:

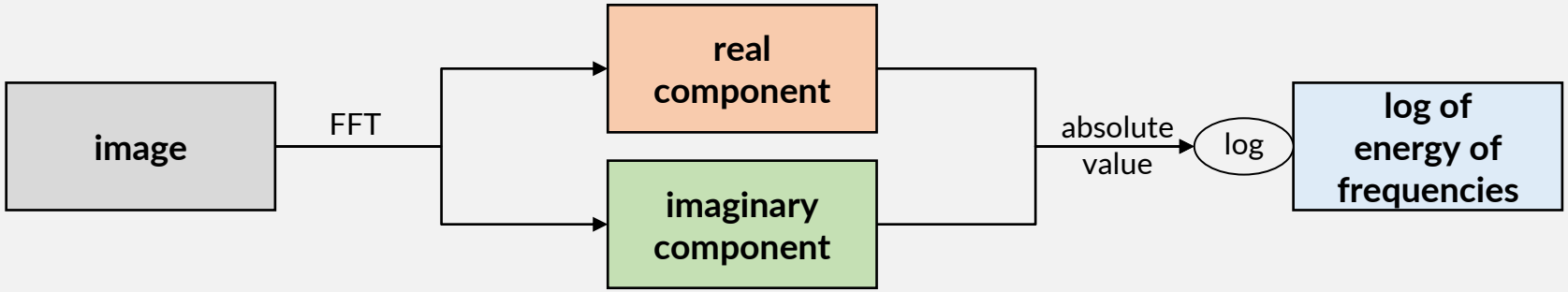
$$\mathbf{G}_{E5L5} = \boldsymbol{v}_{E5}^{\top}\boldsymbol{v}_{L5} = \begin{bmatrix} -1 \\ -2 \\ 0 \\ 2 \\ 1 \end{bmatrix} [1 \quad 4 \quad 6 \quad 4 \quad 1] = \begin{bmatrix} -1 & -4 & -6 & -4 & -1 \\ -2 & -8 & -12 & -8 & -1 \\ 0 & 0 & 0 & 0 & 0 \\ 2 & 8 & 12 & 8 & 2 \\ 1 & 4 & 6 & 4 & 1 \end{bmatrix}$$

  - Since E5L5 and L5E5 measure similar aspects, we combine them into a single kernel and calculate the average of both matrices. By performing these reductions, we obtain 9 kernel matrices:

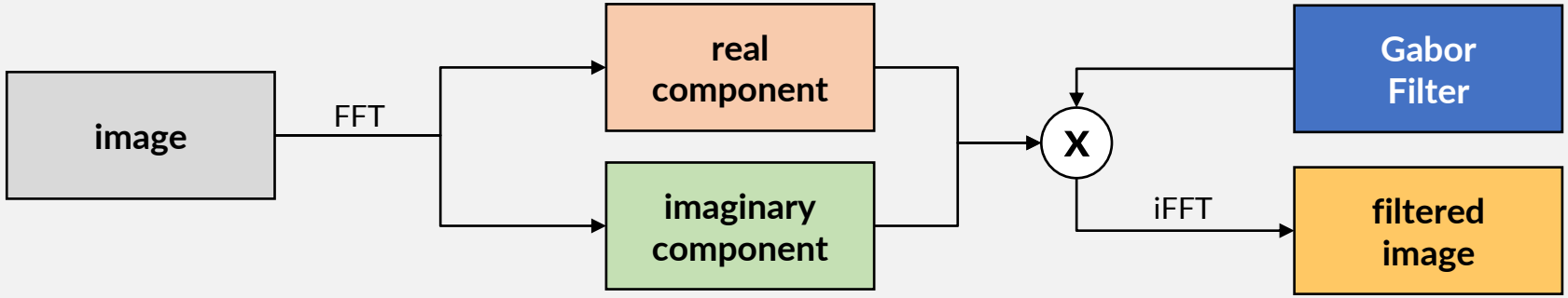
$$\mathbb{G} = \left\{ \frac{\mathbf{G}_{E5L5} + \mathbf{G}_{L5E5}}{2}, \frac{\mathbf{G}_{L5R5} + \mathbf{G}_{R5L5}}{2}, \frac{\mathbf{G}_{E5S5} + \mathbf{G}_{S5E5}}{2}, \frac{\mathbf{G}_{S5L5} + \mathbf{G}_{L5S5}}{2}, \frac{\mathbf{G}_{E5R5} + \mathbf{G}_{R5E5}}{2}, \frac{\mathbf{G}_{S5R5} + \mathbf{G}_{R5S5}}{2},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

  - Using these 9 kernel matrices, we perform convolution to calculate 9 texture energy values, denoted as $e_i(x, y)$ for each pixel, where $1 \leq i \leq 9$. From this point, we can employ the same methods as discussed earlier:
    - **Histograms**: While technically possible, we encounter the challenge of dealing with 9 values per pixel. If we quantize them into 4 ranges, we end up with 4^9 = 262,144 reference energy values. Too many for a meaningful feature. Opting for just 2 ranges results in 2^9 = 512 reference energies, which is more manageable, although it introduces a notable quantification error.
    - **Moments**: For every energy value, we can compute 4 moments and covariance values for the 36 potential pairs. This results in a 72-dimensional feature vector.

- **Gabor Moments** (Fourier approach)
  - The 2D Fourier transform converts a grayscale image into frequency space, producing real and imaginary matrices. To visualize the result, we take the logarithm of the sum of the squares of the real and imaginary parts to reduce large energy differences. The 2D Fast Fourier Transform is a faster version that greatly lowers computational cost. However, it is limited to image dimensions that are powers of two.



  - In Fourier space, we apply Gabor filters to capture different frequency bands and directions. Each Gabor filter is applied to the Fourier-transformed image, which is a complex matrix. The filtered result is converted back to image space by an inverse Fourier transform, for example a fast iFFT. This filtered image shows the presence of the selected frequencies and directions in the original image. With 5 orientations and 3 scales, the filter bank has 15 Gabor filters, producing 15 filtered images. We compute statistical moments for each filtered image to obtain a varied set of texture descriptors. The following pages show the filter banks and their use in Fourier space.



- The Gabor filter is defined as a Gaussian kernel multiplied by a complex sinusoid. Neurophysiological experiments have demonstrated that Gabor filters, when configured with appropriate parameters, exhibit behavior similar to receptive fields in the primary visual cortex. Here's the formal definition:

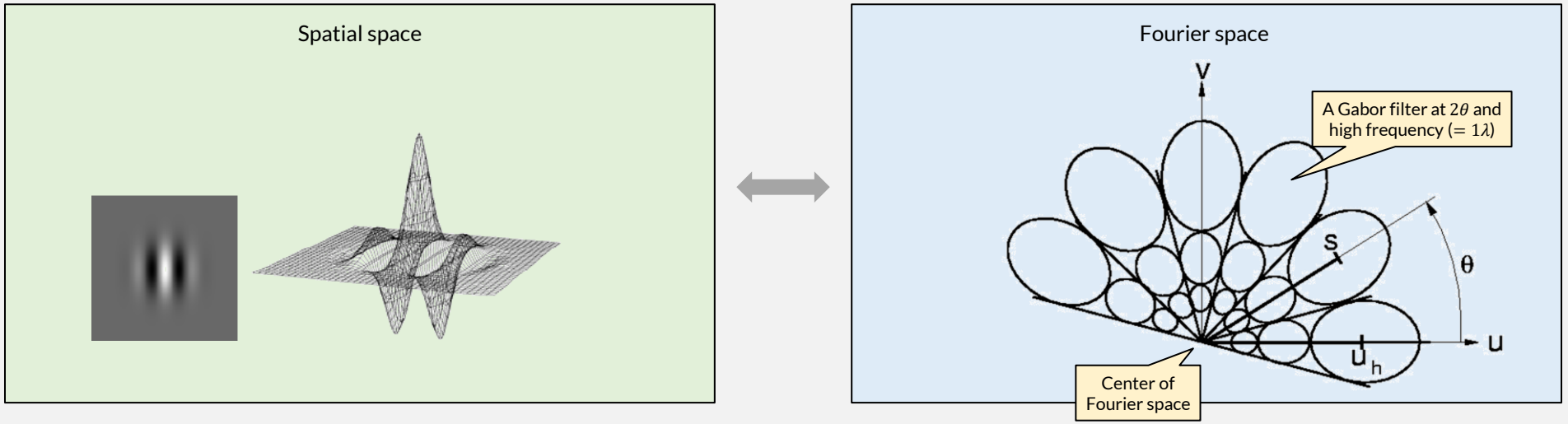
$$g_{\lambda,\theta,\varphi,\sigma,\gamma}(x,y) = e^{-\frac{\bar{x}^2+\gamma^2\bar{y}^2}{2\sigma^2}} \cdot e^{i\left(2\pi\frac{\bar{x}}{\lambda}+\varphi\right)}$$

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

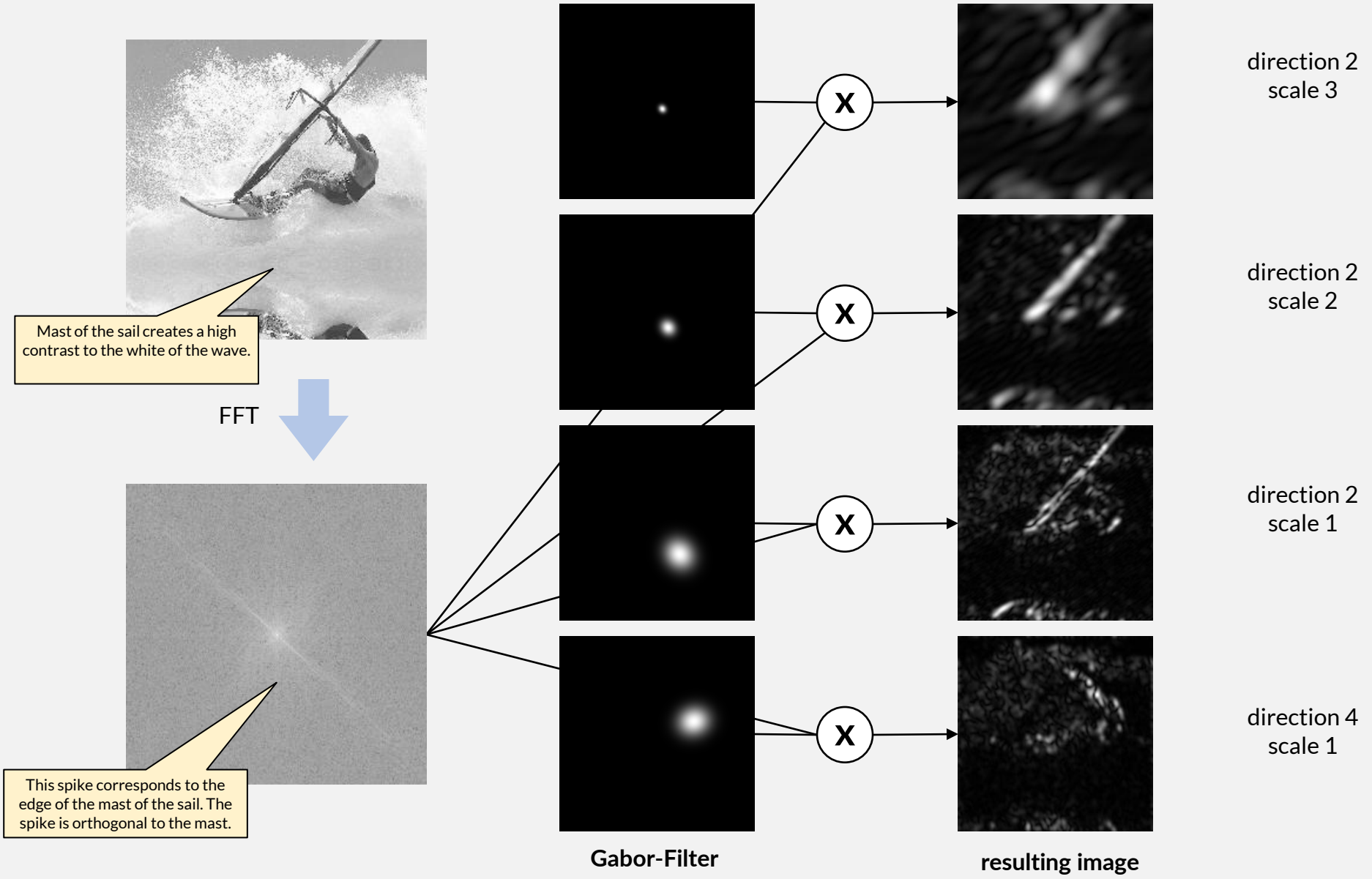
Complex sinusoid with phase φ and wavelength λ. $1/\lambda$ is the frequency of the sinusoid.

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

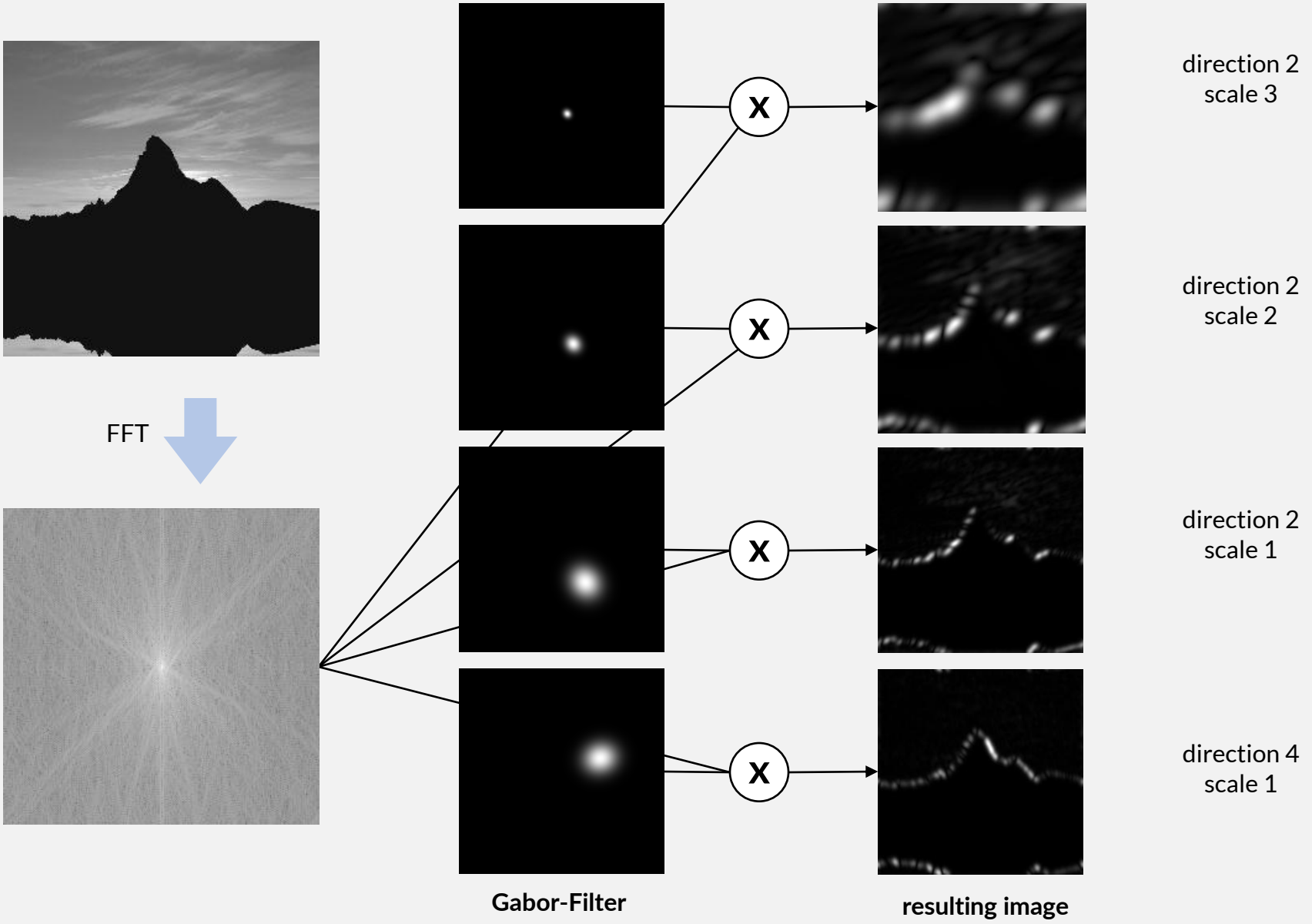
- Before applying the Gaussian and sinusoidal components, the coordinates are rotated by an angle θ. Using this definition and by adjusting the parameters, we can create different filters that are responsive to various frequencies and orientations. When we map the filter bank into the Fourier space, it takes on the layout as shown below on the right side. We can, as in the illustration, apply filters in 6 directions and at 3 different scales, resulting in 18 filters. This is illustrated with the examples on the next pages.
- We can apply Gabor filters both in the spatial and the Fourier space, and we will exploit this fact to accelerate the calculations of texture features. In the spatial space, we obtain Kernels as described at the top and we can use convolution to compute the energy for each direction and scale.



Example (1): Mapping image into Fourier space (FFT) and apply Gabor filter



Example (2): Mapping image into Fourier space (FFT) and apply Gabor filter



- We can apply a Gabor filter in the Fourier domain. However, Fourier transforms can be computationally expensive. The FFT, as an efficient implementation, requires image height and width to be powers of two. Images that do not meet this requirement must be resized or padded carefully to avoid changing orientation or introducing artifacts.
- To avoid these computational challenges, it is more practical to apply Gabor filters by convolving them with the image. This maps each pixel to an energy value determined by the filter's orientation and scale. Because Gabor filters produce complex numbers, we use their absolute values. Many image processing libraries, such as OpenCV and scikit-image, provide built-in Gabor kernels.
- After obtaining the filtered images (as shown in the right-hand columns on the previous pages with the image examples), we can consolidate the results using standard methods like histograms or moments. Usually, we opt for 3-7 directions ($0 \leq \theta \leq \pi$) and 2-5 scales (or frequencies; $1/\lambda$ typically measured in pixels, ranging from 0.05 to 0.5). When working with a large number of filters, moments are preferred to reduce dimensionality and circumvent the complexity of quadratic distance functions.
  - When using moments, we treat the absolute values in the filtered image as the raw data points and calculate mean, variance, skewness, and kurtosis on these values. In addition, we can calculate covariances between filters, adding substantially more features to the description. Let $\tilde{f}_i(x,y)$ represent the filtered (complex) image representation after applying the i-th Gabor filter. We get:

$$\mu_i = \frac{1}{N \cdot M} \sum_{x,y} |\tilde{f}_i(x,y)| \qquad v_i = \frac{1}{N \cdot M} \sum_{x,y} \left(|\tilde{f}_i(x,y)| - \mu_i\right)^2 \qquad s_i = \frac{1}{N \cdot M} \sum_{x,y} \left(\frac{|\tilde{f}_i(x,y)| - \mu_i}{\sqrt{v_i}}\right)^3$$

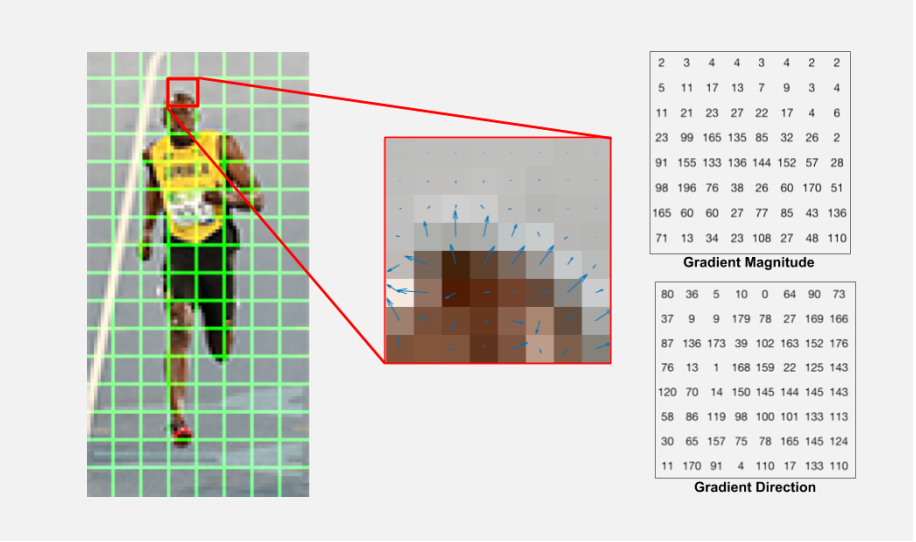
$$k_i = \frac{1}{N \cdot M} \sum_{x,y} \left(\frac{|\tilde{f}_i(x,y)| - \mu_i}{\sqrt{v_i}}\right)^4 \qquad cov_{i,j} = \frac{1}{N \cdot M} \sum_{x,y} \left(|\tilde{f}_i(x,y)| - \mu_i\right) - \left(|\tilde{f}_j(x,y)| - \mu_j\right)$$

  - Example: choosing 5 directions and 3 scales leads to
    - 15 mean, 15 variance, 15 skewness, and 15 kurtosis values
    - 105 covariance values
    - Total: 165 values

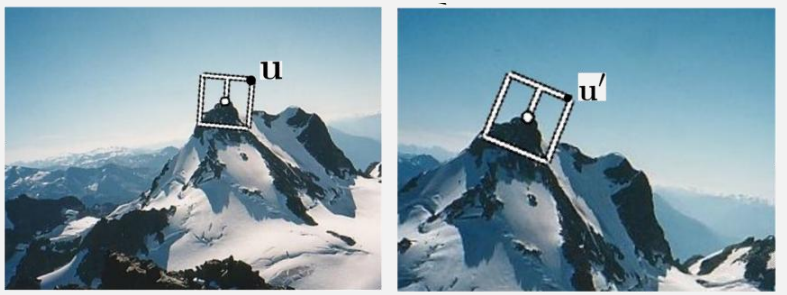
11.4 Shape Information

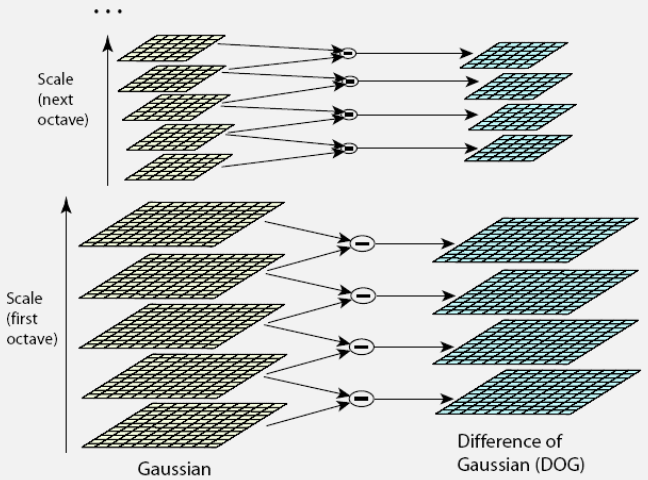
- Shape information is typically more important in object databases where objects are described by their shape, e.g., a 2D or 3D model. In images, we have different approaches to extract shape information:
  - **Edge Detection**: Edge detection algorithms like the Canny edge detector can locate the boundaries of objects in an image, which is a fundamental step in shape analysis.
  - **Contour Detection**: Contour detection algorithms, such as the OpenCV's `findContours` function, identify continuous curves that represent the shapes of objects within an image.
  - **Region-based Segmentation:** Techniques like region growing and region splitting can be used to group pixels with similar properties, which helps identify objects and their shapes.
- Once shape information is available, we can use methods similar to those for texture analysis to characterize the contour in terms of edge directions or extracted shapes. Most techniques assume that we can describe the object's area or boundary in the image, allowing us to calculate the following shape descriptors:
  - **Area**: The percentage of pixels within the segment relative to the entire image.
  - **Centroid**: The average x and y-values of the region (in the absence of mass values).
  - **Axis of Least Inertia**: This axis minimizes the squared distances to the region's boundary, useful for normalizing regions into a primary direction.
  - **Eccentricity**: The ratio of length to width of a bounding box in the principal direction.
  - **Circularity Ratio**: Indicates how closely the shape resembles a circle, often defined as the ratio of the area of the smallest containing circle to the region's area.
- HOG (Histogram of Oriented Gradients) and SIFT (Scale-Invariant Feature Transform) are advanced computer vision techniques that capture and describe shape-related features in images. HOG quantifies gradient orientations in local image regions, making it suitable for object detection and recognizing shapes by analyzing the distribution of gradient directions. In contrast, SIFT identifies distinctive key points in an image, including corners and junctions, and describes local structures, including shapes. SIFT is valuable for shape recognition and matching due to its scale-invariance and robustness to variations in object size. Both methods are essential tools in computer vision for extracting and characterizing shape-related information, with HOG focusing on global shape analysis and object detection, while SIFT specializes in local shape descriptions and key point matching.

- The **Histogram of Oriented Gradients (HOG)** technique was initially introduced in 1986 and experienced a resurgence in popularity when Dalal and Triggs utilized it in 2005 for pedestrian detection. Since then, it has been expanded and frequently serves as input data for neural networks.
  - **Step 1** computes gradients, for example by applying Sobel operators to a grayscale image. HOG uses unsigned gradients, so directions range from 0 to π (0 to 180 degrees). Angles between π and 2π are reduced by subtracting π. Some HOG implementations let you choose signed or unsigned gradients, but Dalal and Triggs found the unsigned option worked best for pedestrian detection.
  - In **Step 2** the image is split into cells, each covering an 8x8 pixel area. For each cell, HOG computes a histogram with 9 bins, a number found to work best for this case. The histogram is built from the gradient directions of the cell's 64 pixels, with each direction weighted by its gradient magnitude, as shown in the image below.
  - In **Step 3**, gradient magnitudes, which can vary with illumination, need to be normalized before comparing histograms. HOG uses a technique where it combines four neighboring cells and normalizes the concatenated histograms, resulting in a 36-bin histogram that sums up to 1. These four neighboring cells are organized in 2x2 cell groups, each covering an 8x8 pixel area, and they are shifted along the image at 8-pixel intervals. Each block produces a normalized histogram with 36 bins, and these blocks have partial overlaps.
  - In **Step 4**, you can either combine the histograms into global features or maintain a "bag" of local features for searching.
  - The HOG descriptors are used to train machine learning models, such as Support Vector Machines (SVMs) or neural networks, for object detection.
  - HOG features are widely used for object detection, human detection in surveillance, gesture recognition, face detection, biometric identification, and other tasks. Because they capture shape and structural information, they are an essential tool in many computer vision applications.

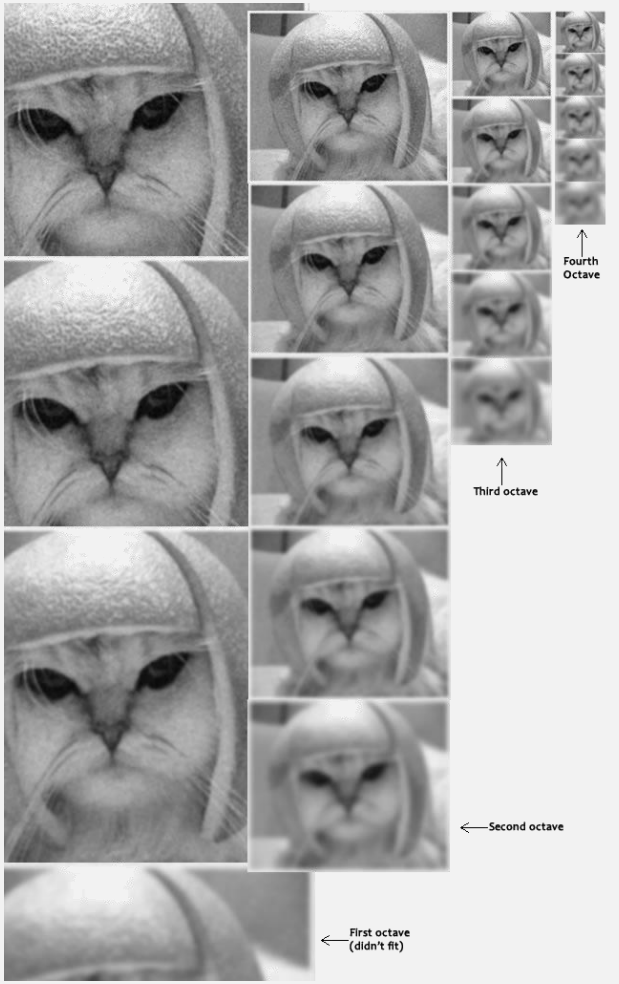


- The **Scale Invariant Feature Transform (SIFT)** method robustly extracts features that can successfully match even when there are substantial changes in viewpoint. Take a look at the images in the upper right corner: SIFT is capable of matching key points, such as the mountain top, even in the presence of substantial alterations in scale, rotation, and viewpoint. The algorithms works in 4 steps:
  - **Step 1:** We generate an image pyramid using Gaussian filters at various standard deviations (σ) and scales, referred to as "octaves" in SIFT terminology (see right side for illustration). Each octave is downsampled to one-fourth of the previous one. Within each octave, the image is progressively blurred using Gaussian filters with increasing σ values.
    - In every octave, we compute the difference of Gaussians (DOG) by subtracting neighboring images. These DOG images act as edge detectors for specific frequency bands.
    - The DOG image pyramid contains potential edges and points of interest, identified as the local minima and maxima within the DOG images. See illustration below.

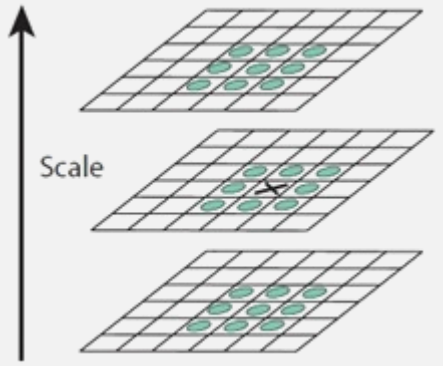




Step 1



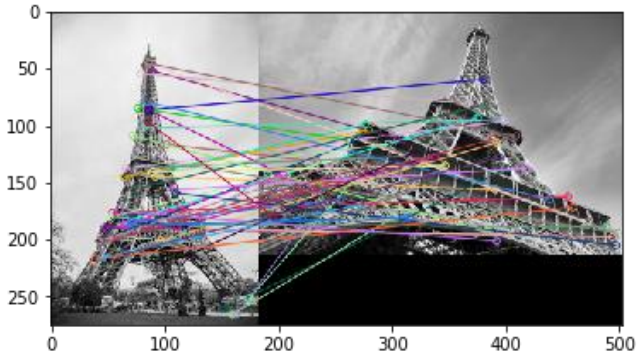
- **Step 2:** We identify interesting points in the DOG pyramid using a one-pixel neighborhood. In the picture on the right, where "x" represents the current pixel, there are 8 neighbors in the same plane and 9 neighbors from the plane above and below. If the pixel is found to be a local minima or maxima within this neighborhood, it is marked accordingly. Otherwise, the pixel is discarded.
  - Starting with five Gaussian-blurred images in each octave, we create four DOG images, resulting in two extrema images at each octave. To reduce the number of keypoints, we eliminate all pixels with DOG values below a specified threshold, as these represent "flat" points. We also filter out edge points by considering their gradients; edges have a large gradient perpendicular to the edge and a small gradient along the edge. Our interest lies in corner points characterized by two substantial gradients.

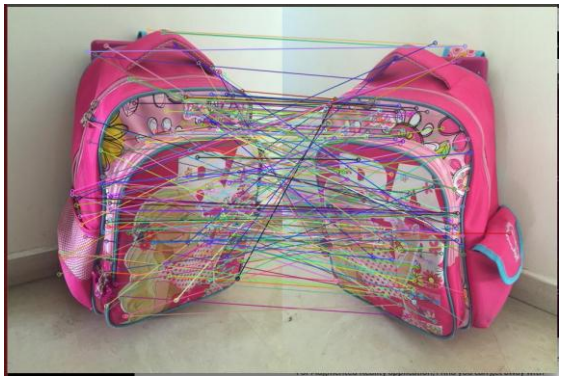


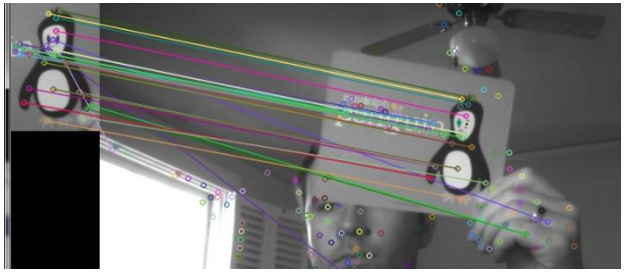
Step 2

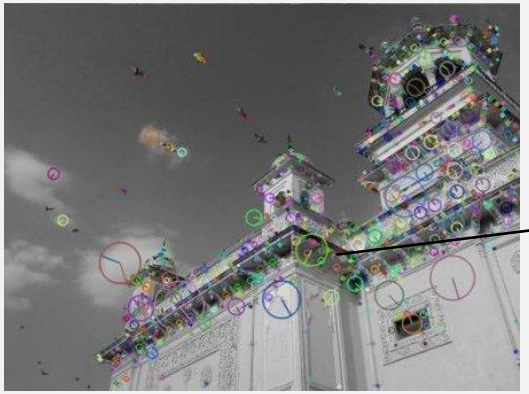
- **Step 3:** To create a rotation-invariant feature, we determine the principal orientation for each keypoint. SIFT builds a local histogram of gradient directions within the neighborhood of the keypoint. The size of this neighborhood window scales with the keypoint's scale. Each gradient direction is added to the histogram, weighted by its magnitude. The bin in the histogram with the highest value corresponds to the dominant direction. In the case of ties, all directions are considered.
  - SIFT uses the dominant orientation to normalize feature extraction, as shown in the next step. If it finds multiple orientations, SIFT creates features for each one. This normalization makes it easy to compare keypoints found from different viewpoints using a simple metric.
  - The primary direction of the keypoint may not align with its gradient direction.
- **Step 4**: SIFT places a 4x4 grid oriented in the dominant direction around the keypoint, with the grid size scaling according to the keypoint's scale. Within each grid cell, a finer 4x4 mesh defines a neighborhood, and an 8-bin histogram captures gradient directions. For each point in this finer mesh, gradient orientation and magnitude are calculated, and the direction is added to the histogram with a Gaussian weight based on its distance from the keypoint. This process results in 8 values for each cell within the larger 4x4 grid, yielding a total of 128 feature values. See the bottom of the next page for an illustration of this step.

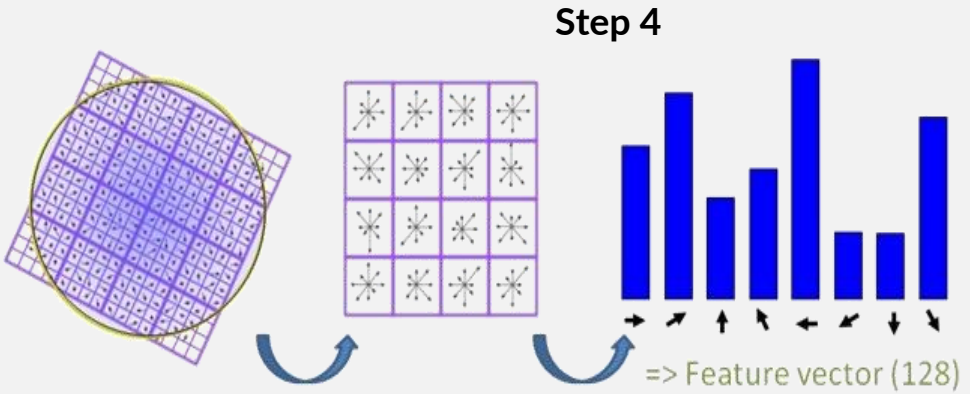
– SIFT features are invariant to scale, translation, and rotation. They are modeled on receptive fields in the primary visual cortex and capture local image patterns by orientation. SIFT descriptors are highly distinctive, and even small objects can produce many descriptors. Although they are complex, SIFT features can be computed nearly in real time. They are widely used for object recognition, motion detection, image alignment, and image stitching. Implementations exist in OpenCV, and scikit-image provides related methods such as daisy and harris. Like HOG, SIFT descriptors can be used as input to machine learning models.











11.5 Literature and Links

Image Features

- 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
- 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