

COMPUTER VISION

1.1 Introduction to Computer Vision

Computer vision is a field of artificial intelligence (AI) that enables computers to interpret and make decisions based on visual data. Unlike humans, who perceive the three-dimensional world effortlessly, computers require mathematical models and algorithms to process images and recognize objects.

➤ Human Vision vs. Computer Vision

- Human vision is complex and intuitive, allowing us to recognize objects, emotions, and scenes instantaneously.
- Computer vision aims to replicate human perception by using algorithms and deep learning models to analyse images and videos.
- The challenge lies in converting raw pixel data into meaningful insights.

➤ Challenges in Computer Vision

- **Inverse Problem:** Involves reconstructing three-dimensional scenes from two-dimensional images.
- **Physics-Based Models:** Understanding light reflection, shading, and transparency.
- **Machine Learning Dependency:** Requires large datasets to train models effectively.
- **Error Proneness:** Unlike humans, computer vision algorithms often misinterpret images due to variations in lighting, occlusions, and object distortions.

➤ Applications of Computer Vision

Computer vision has widespread applications across various industries. Some key areas include:

Industrial Applications

- **Optical Character Recognition (OCR):** Used in postal systems for reading handwritten text and number plate recognition.
- **Machine Inspection:** Quality assurance in manufacturing, defect detection in materials.
- **Retail Automation:** Self-checkout systems and automated stores.
- **Warehouse Logistics:** Autonomous robots for sorting and package delivery.
- **Medical Imaging:** Assists in diagnosing diseases and tracking brain morphology over time.
- **Self-Driving Vehicles:** Autonomous navigation using computer vision for real-time object detection.
- **3D Model Building (Photogrammetry):** Constructing 3D models from drone and aerial photography.
- **Match Move in Film Industry:** Used for integrating CGI with real-world footage.
- **Motion Capture (MoCap):** Capturing human movements for animation and virtual reality.

- Surveillance: Security monitoring, traffic analysis, and public safety.
- Fingerprint and Facial Recognition: Biometric authentication and forensic applications.

Figure 1.3 some common optical illusions and what they might tell us about the visual system: (a) the classic where the lengths of the two horizontal lines appear different, probably due to the imagined perspective effects. (b) The “white” square B in the shadow and the “black” square A in the light actually have the same absolute intensity value. The percept is due to brightness constancy, the visual system’s attempt to discount illumination when interpreting colors. Image courtesy of Ted Adelson. (c) A variation of the Hermann grid illusion, courtesy of Hany Farid. As you move your eyes over the figure, gray spots appear at the intersections. (d) Count the red Xs in the left half of the figure. Now count them in the right half. Is it significantly harder? The explanation has to do with a pop-out effect, which tells us about the operations of parallel perception and integration pathways in the brain.

Consumer-Level Applications

- Photo Stitching: Creating panoramas by merging overlapping images.
- Exposure Bracketing: Combining multiple exposures to optimize image lighting.
- Morphing: Transforming one image into another with smooth transitions.
- 3D Modelling: Converting images into 3D representations.
- Video Stabilization: Reducing motion blur and shake in recorded videos.
- Photo-Based Walkthroughs: Virtual navigation through image datasets.

- Face Detection & Visual Authentication: Used in smart cameras and device security.

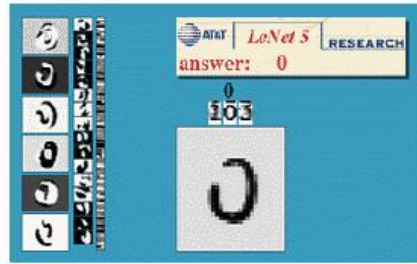
➤ **Problem-Solving Approach in Computer Vision**

A structured approach is essential in designing computer vision applications:

- Problem Definition: Clearly define the objective and constraints.
- Technique Selection: Choose appropriate algorithms and models.
- Implementation & Evaluation: Test models on synthetic and real-world data.
- Machine Learning Considerations: Ensure unbiased and representative training data.
- Scientific Approach: Incorporate physics and statistical models to improve accuracy.
- Bayesian Modelling: Use probabilistic models to improve inference and decision-making.

➤ **Algorithmic Approach in Computer Vision**

- Computer vision often involves solving inverse problems and estimating unknown quantities.
- Algorithms should be robust to noise and adaptable to real-world conditions.
- Efficient use of computational resources is crucial.
- Bayesian techniques help ensure robust estimations.
- Efficient search, minimization, and linear system-solving methods improve performance.
- Many algorithms are high-level and require students to implement specific steps.



(a)



(b)



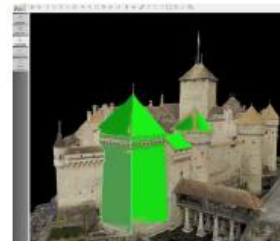
(c)



(d)



(e)



(f)

Figure 1.4 some industrial applications of computer vision: (a) optical character recognition (OCR), (b) mechanical inspection (c) warehouse picking, (d) medical imaging, (e) self-driving cars, (f) drone-based photogrammetry.

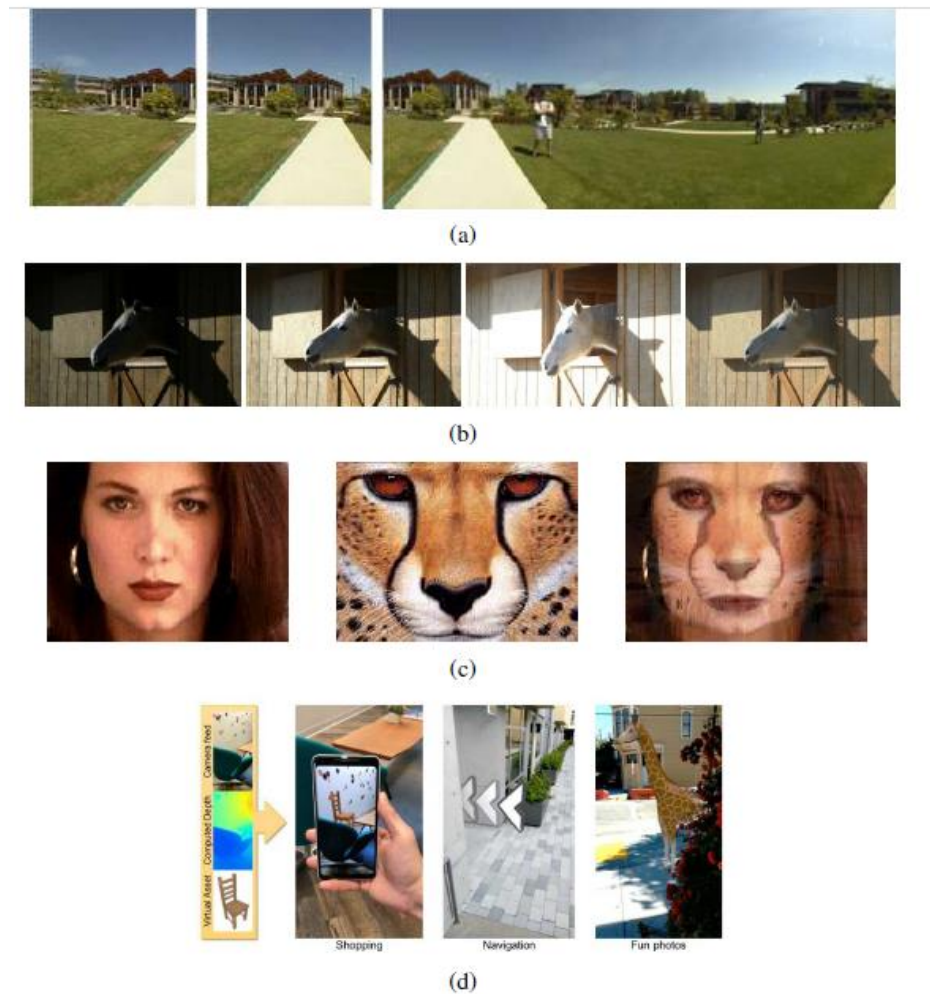


Figure 1.5 some consumer applications of computer vision: (a) image stitching: merging different views. (b) Exposure bracketing: merging different exposures; (c) morphing: blending between two photographs (d) smartphone augmented reality showing real-time depth occlusion effects.

➤ Future of Computer Vision

- Advancements in deep learning and AI continue to enhance accuracy and real-world applicability.
- Integration with robotics, augmented reality (AR), and the Internet of Things (IoT) is expanding computer vision's capabilities.
- Ethical considerations, including privacy and bias in AI models, are crucial for responsible deployment.

1.2 A Brief History of Computer Vision

Introduction: Computer vision, as a field of study, has evolved significantly over the past five decades. This section presents a chronological overview of key developments in the field, focusing on pioneering research, major breakthroughs, and the technological advancements that have shaped modern computer vision.

1970s: The Birth of Computer Vision

Computer vision emerged in the early 1970s as a subfield of artificial intelligence (AI) and robotics. The initial goal was to develop visual perception systems capable of mimicking human vision, enabling robots to interact intelligently with their surroundings.

- **Early Ambitions and Challenges:** Researchers at institutions like MIT, Stanford, and Carnegie Mellon University (CMU) believed that solving the problem of visual input processing was a stepping stone toward achieving advanced reasoning and planning capabilities.
- **Historical Anecdote:** In 1966, Marvin Minsky at MIT tasked undergraduate student Gerald Jay Sussman with a summer project: linking a camera to a computer to describe what it saw. This underestimated the complexity of the task, revealing the vast challenges in replicating human vision computationally.
- **Differentiation from Image Processing:** Unlike digital image processing, which focused on manipulating images (e.g., filtering, enhancement), computer vision aimed to reconstruct the three-dimensional (3D) structure of a scene from two-dimensional (2D) images.
- **Early Approaches:**
 - **Edge Detection & Scene Understanding:** Techniques such as line labeling (Huffman, 1971; Clowes, 1971; Waltz, 1975) and edge-based 3D reconstruction (Roberts, 1965) laid foundational work.
 - **Shape Representation:** Researchers explored ways to model 3D structures using geometric primitives, such as generalized cylinders (Agin & Binford, 1976; Nevatia & Binford, 1977).
 - **Understanding Light & Shading:** Efforts to interpret shading and lighting variations in images led to the concept of intrinsic images (Barrow & Tenenbaum, 1981) and the 2.5D sketch model (Marr, 1982).

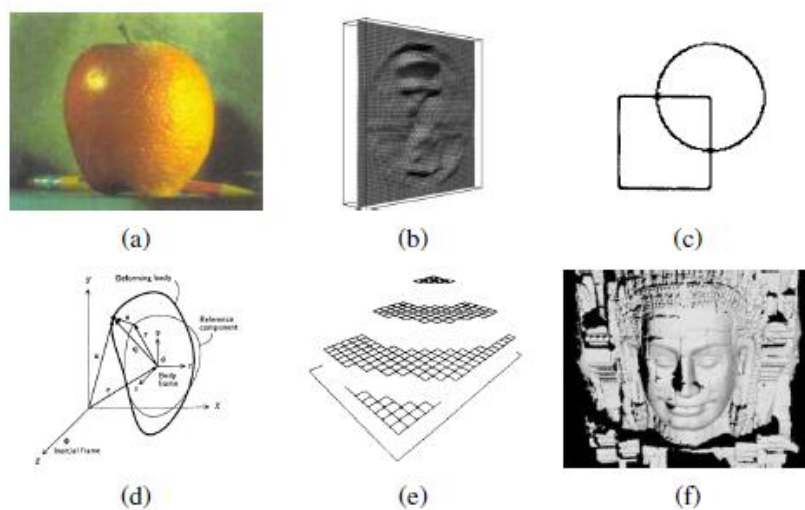


Figure 1.8 Examples of computer vision algorithms from the 1980s: (a) pyramid blending. (b) shape from shading. (c) edge detection. (d) physically based models (e) regularization-based reconstruction. (f) range data acquisition and merging.

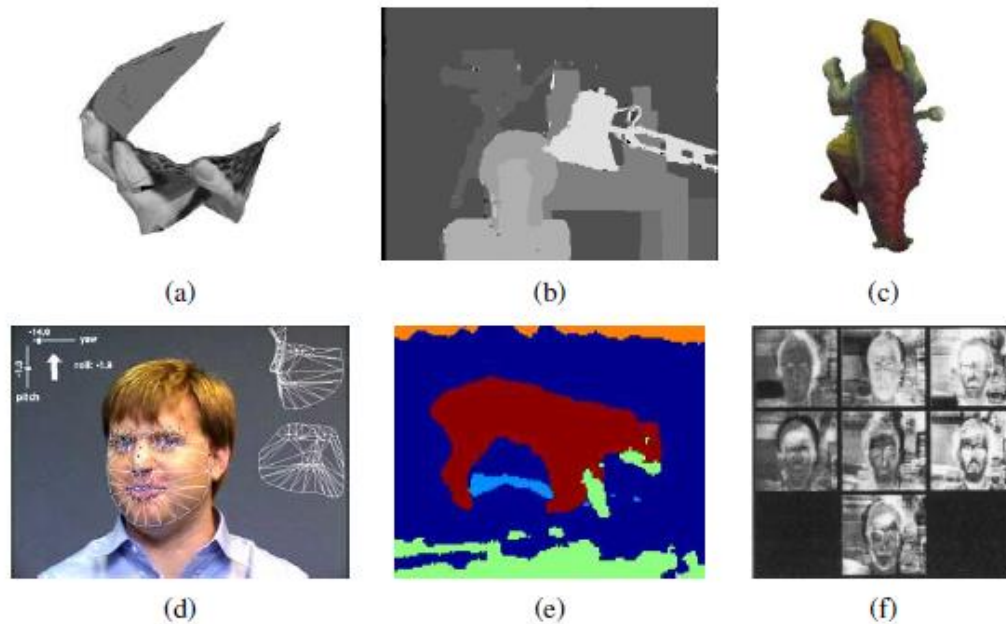


Figure 1.9 Examples of computer vision algorithms from the 1990s: (a) factorization-based structure from motion (b) dense stereo matching (c) multi-view reconstruction (d) face tracking (e) image segmentation

➤ 1980s: Mathematical Rigor and Computational Techniques

The 1980s witnessed the adoption of sophisticated mathematical models for quantitative image and scene analysis.

- **Multi-Resolution Image Analysis:** Image pyramids (Burt & Adelson, 1983) were developed for coarse-to-fine image processing, leading to the emergence of scale-space representations (Witkin, 1983).
- **Shape-from-X Techniques:** Various methods were introduced to infer 3D shapes from different visual cues:
 - Shape from Shading: (Horn, 1975; Pentland, 1984)
 - Photometric Stereo: (Woodham, 1981)
 - Shape from Texture: (Witkin, 1981; Malik & Rosenholtz, 1997)
 - Shape from Focus: (Nayar et al., 1995)
- **Edge Detection Improvements:** The Canny Edge Detector (Canny, 1986) became a standard technique for detecting image contours.
- **Optimization-Based Methods:**
 - Variational optimization and regularization methods were used to unify stereo vision, optical flow, and shape-from-X algorithms (Terzopoulos, 1983; Poggio, Torre, & Koch, 1985).
 - Markov Random Fields (MRFs) were introduced for structured image analysis (Geman & Geman, 1984).
- **3D Range Data Processing:** Techniques for acquiring and reconstructing 3D models were actively explored (Besl & Jain, 1985; Faugeras & Hebert, 1987).

➤ 1990s: Algorithmic Advancements and Statistical Learning

The 1990s saw the emergence of more advanced computational techniques, making computer vision more practical for real-world applications.

- Structure from Motion (SfM): Methods to reconstruct 3D scenes from moving cameras gained prominence (Faugeras, 1992; Hartley & Zisserman, 2004).
- Statistical Learning Approaches:
 - Eigenfaces (Turk & Pentland, 1991) revolutionized face recognition using Principal Component Analysis (PCA).
 - Hidden Markov Models (HMMs) and Kalman filters were applied to motion tracking and object recognition.
- Segmentation & Graph-Based Approaches:
 - The Normalized Cuts method (Shi & Malik, 2000) improved image segmentation techniques.
 - Graph-cut algorithms (Boykov, Veksler, & Zabih, 1998) optimized stereo correspondence.
- Computer Vision & Graphics Integration:
 - Image-based rendering techniques enabled applications like panoramic stitching (Mann & Picard, 1994) and morphing (Beier & Neely, 1992).

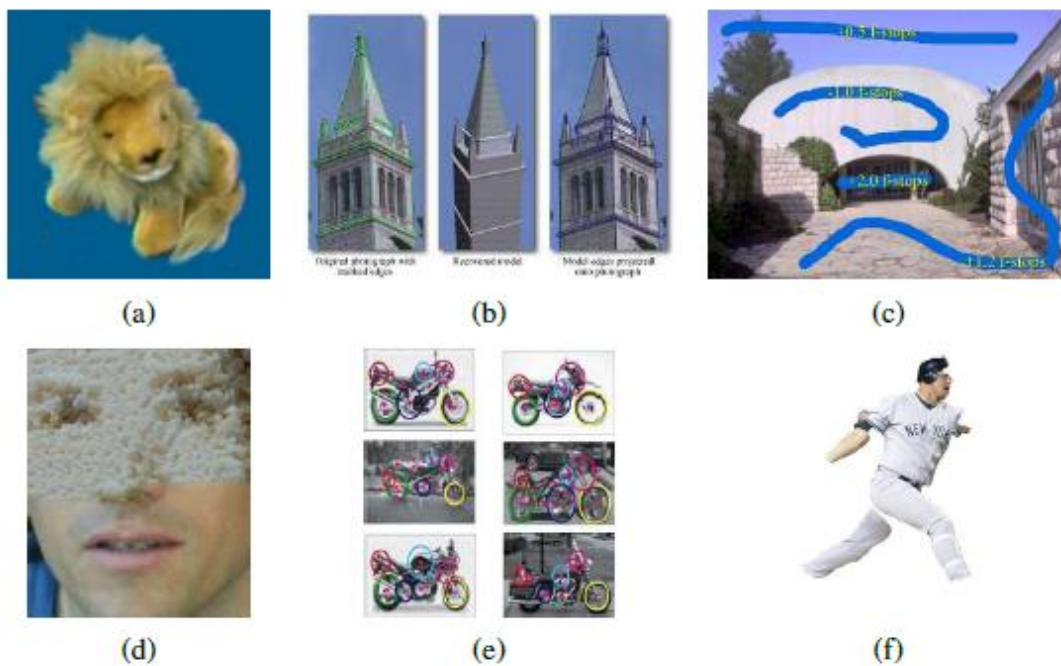


Figure 1.10 Examples of computer vision algorithms from the 2000s: (a) image-based rendering (b) image-based modeling (c) interactive tone mapping (d) texture synthesis, (e) feature-based recognition (f) region-based recognition

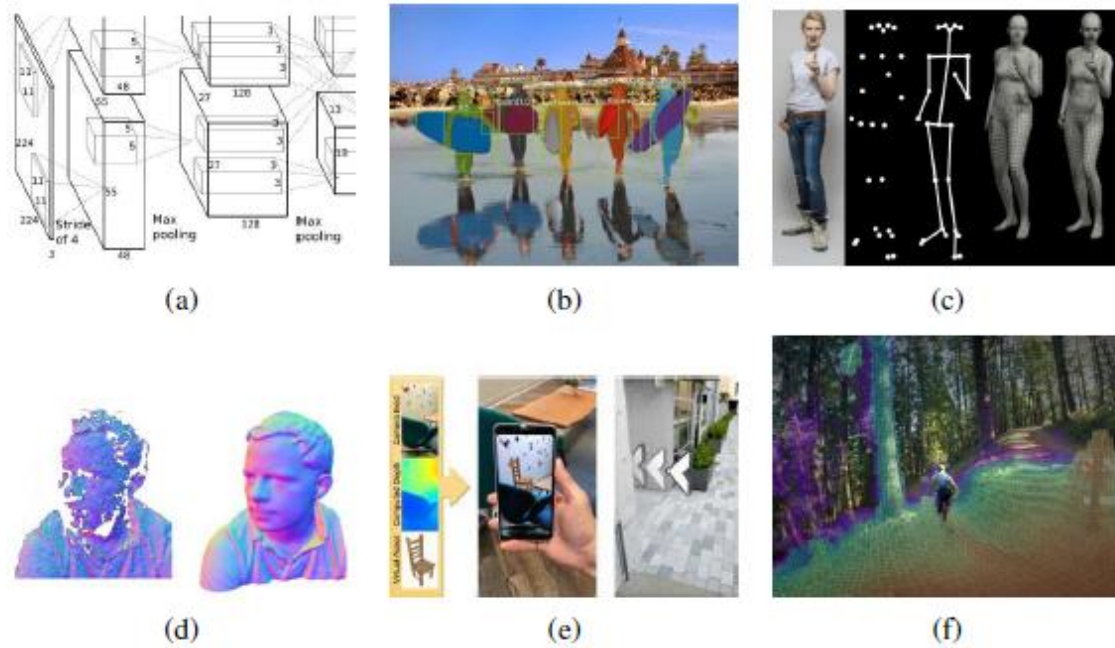


Figure 1.11 Examples of computer vision algorithms. (a) the Supervision deep neural network (b) object instance segmentation (c) whole body, expression, and gesture fitting from a single image. (d) fusing multiple color depth images using the KinectFusion real-time system (e) smartphone augmented reality with real-time depth occlusion effects. (f) 3D map computed in real-time on a fully autonomous.

➤ 2000s: The Rise of Machine Learning and Large-Scale Data

In the 2000s, computer vision benefited from increased computational power, larger datasets, and improved machine learning models.

- Feature-Based Recognition:
 - Scale-Invariant Feature Transform (SIFT) (Lowe, 1999) became a standard feature extraction technique.
 - Speeded-Up Robust Features (SURF) (Bay et al., 2006) improved efficiency in object detection.
- Deep Learning Emergence:
 - Convolutional Neural Networks (CNNs) started gaining traction for tasks like object recognition.
- Advancements in 3D Vision:
 - Multi-view stereo algorithms improved 3D reconstruction capabilities (Seitz et al., 2006).
 - Techniques like Structure from Motion (SfM) and Simultaneous Localization and Mapping (SLAM) became more refined.
- Real-World Applications:
 - Facial recognition, autonomous vehicles, and augmented reality (AR) saw significant advancements, powered by more robust vision models.

So the evolution of computer vision has been marked by ground-breaking theories, mathematical advancements, and increasing computational capabilities. From early edge-detection methods to modern deep learning-based recognition

systems, computer vision continues to revolutionize industries, including robotics, healthcare, surveillance, and entertainment. The field is expected to grow further, driven by AI advancements and the availability of large-scale visual datasets.

2.2 Photometric Image Formation

In the process of modeling image formation, we analyze how three-dimensional (3D) geometric features of the real world are projected onto a two-dimensional (2D) image. However, an image is not just a collection of 2D shapes; rather, it is composed of discrete color or intensity values.

This leads to important questions:

- What determines these color or intensity values?
- How do factors such as environmental lighting, surface properties, object geometry, camera optics, and sensor characteristics influence them?

To answer these questions, we establish a set of models that describe these interactions and explain the generative process of image formation. More comprehensive discussions on this topic can be found in textbooks related to computer graphics and image synthesis (Cohen & Wallace, 1993; Sillion & Puech, 1994; Watt, 1995; Glassner, 1995; Weyrich et al., 2009; Hughes et al., 2013; Marschner & Shirley, 2015).

2.2.1 Lighting and Its Role in Image Formation

Light is fundamental for image creation. Without illumination, a scene remains dark, and no image can be captured. The process of image formation depends on the presence of one or more light sources. While some imaging techniques, such as fluorescence microscopy and X-ray tomography, operate differently, the focus here is on conventional image formation.

Types of Light Sources:

1. **Point Light Sources:**
 - Originate from a single location in space (e.g., a small light bulb or the Sun).
 - If the light source is at a significant distance (like the Sun), it may be approximated as an infinite source.
 - The intensity of light from a point source decreases proportionally to the square of the distance from the illuminated object, as light disperses over a larger spherical area.
 - In some cases, point sources exhibit directional variations, but for simplification, this aspect is often ignored.
2. **Area Light Sources:**
 - Unlike point sources, these cover a finite region and emit light across a broad area (e.g., a fluorescent ceiling light).
 - A simple example is a rectangular fluorescent panel that emits light uniformly in all directions.
 - When light distribution is more directional, a four-dimensional light field representation is used (Ashdown, 1993).
3. **Environment Maps and Complex Light Distributions:**
 - Certain real-world lighting conditions, such as those experienced by an object outdoors, are better represented using an **environment map**.

- Environment maps record incident light from all directions and map it to corresponding color values.
- These maps assume that all light sources exist at an infinite distance.
- They can be represented in different formats:
 - Cubical face maps (Greene, 1986)
 - Longitude–latitude maps (Blinn & Newell, 1976)
 - Reflective sphere images (Watt, 1995)
- A practical way to create an environment map is by capturing an image of a mirrored sphere and then processing it to match the required mapping (Debevec, 1998).

2.2.2 Reflectance and Shading

When light interacts with an object's surface, it undergoes scattering and reflection. These interactions define how the object appears in an image. Several models exist to describe these behaviors:

1. **Bidirectional Reflectance Distribution Function (BRDF):**
 - A general function that describes how light is reflected at an interface given the incoming and outgoing directions of light.
 - It accounts for both diffuse and specular reflections.
2. **Diffuse Reflection:**
 - Occurs when light scatters uniformly in all directions upon hitting a rough surface.
 - The Lambertian reflectance model assumes that the apparent brightness of the surface remains constant regardless of the viewing angle.
3. **Specular Reflection:**
 - Happens when light reflects in a particular direction, as seen in smooth and shiny surfaces (e.g., mirrors, polished metals).
 - This type of reflection creates highlights and is angle dependent.
4. **Phong Reflection Model:**
 - A more practical approach that combines diffuse and specular components.
 - The model introduces a shininess factor that controls the sharpness of the specular highlight.
5. **Global Illumination Considerations:**
 - Beyond direct lighting, real-world scenes involve multiple light interactions such as:
 - **Shadowing:** When objects block light and create dark regions.
 - **Interreflection:** When light bounces off surfaces before reaching the camera.
 - **Caustics:** When light is focused through reflective or transparent surfaces (e.g., water or glass patterns on the floor).

Understanding these principles allows for the development of realistic computer-generated images, improved computer vision algorithms, and enhanced image processing techniques. By accurately modeling lighting and

reflectance, we can create more visually accurate and physically meaningful representations of the world in digital images.

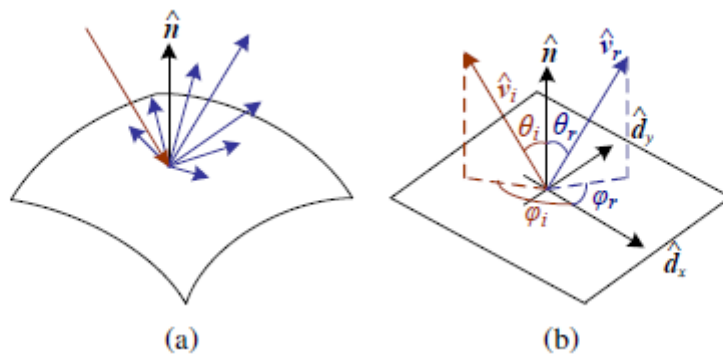


Figure 2.15 (a) Light scatters when it hits a surface. (b) The bidirectional reflectance distribution function (BRDF) $f(\theta_i, \phi_i, \theta_r, \phi_r)$ is parameterized by the angles that the incident, $\hat{v}_i$, and reflected, $\hat{v}_r$, light ray directions make with the local surface coordinate frame ($\hat{d}_x, \hat{d}_y, \hat{n}$).

➤ The Bidirectional Reflectance Distribution Function (BRDF)

When light interacts with a surface, it does not simply bounce off in a uniform manner. Instead, the way light is scattered and reflected depends on the material properties of the surface. The **Bidirectional Reflectance Distribution Function (BRDF)** is a mathematical model that describes this behavior.

Definition and Purpose of BRDF

The BRDF is a **four-dimensional function** that defines how much light arriving from one direction (incident light) is reflected in another direction. It is typically represented as:

$$f_r(\theta_i, \phi_i, \theta_r, \phi_r, \lambda)$$

Where:

- θ_i and ϕ_i are the angles of the incident light relative to the surface.
- θ_r and ϕ_r are the angles of the reflected light relative to the surface.
- λ represents the wavelength of light, since different wavelengths (colors) may be reflected differently.

In simpler terms, the BRDF describes how much light is reflected in a given direction based on how the light originally struck the surface. This function is crucial for **realistic rendering** in computer graphics, image processing, and physics-based simulations of light interactions.

Helmholtz Reciprocity and BRDF Symmetry

One of the fundamental properties of the BRDF is **reciprocity**, also known as **Helmholtz reciprocity**. This principle states that if we swap the incident and reflected light directions, the BRDF remains unchanged. Mathematically, this means:

$$f_r(\theta_i, \phi_i, \theta_r, \phi_r, \lambda) = f_r(\theta_r, \phi_r, \theta_i, \phi_i, \lambda)$$

This property arises due to the **physical laws of light transport** and ensures consistency in how light behaves across different viewing angles.

Isotropic vs. Anisotropic Surfaces

Surfaces can generally be classified based on how they reflect light:

1. Isotropic Surfaces

- These surfaces reflect light **uniformly** in all directions, meaning there is no preferred orientation for light transport.
- Examples include **plastic, unpolished wood, and matte surfaces**.
- The BRDF for isotropic surfaces simplifies to:

$$f_r(\theta_i, \theta_r, |\phi_r - \phi_i|, \lambda)$$

□

This means that reflection depends only on the angle difference between the incident and reflected directions, rather than their absolute orientations.

2. Anisotropic Surfaces

- These surfaces have **directional dependence**, meaning light reflection varies based on the orientation of surface features.
- A common example is **brushed aluminum or scratched metal**, where the scratches create a preferred direction of reflection.
- The BRDF of anisotropic materials must account for this directional dependence and cannot be simplified in the same way as isotropic materials.

➤ Alternative Representations of BRDF

- Instead of using explicit angles, the BRDF can also be expressed in terms of **unit direction vectors**:

$$f_r(\hat{\mathbf{v}}_i, \hat{\mathbf{v}}_r, \hat{\mathbf{n}}, \lambda)$$

Where:

- $\hat{\mathbf{v}}_i$ is the incident light direction.
- $\hat{\mathbf{v}}_r$ is the reflected light direction.
- $\hat{\mathbf{n}}$ is the normal vector to the surface.

This formulation is particularly useful in **computer graphics and physically based rendering (PBR)** since these vectors can be computed directly from 3D models.

Key Points:

- The **BRDF** models how light is reflected from a surface based on its material properties and incoming light direction.
- It is a **four-dimensional function** that depends on incident and reflected angles, as well as the light wavelength.
- **Helmholtz reciprocity** states that swapping incident and reflected directions does not change the BRDF value.
- **Isotropic surfaces** reflect light evenly, while **anisotropic surfaces** have a directional dependence.
- BRDFs are essential in **computer graphics, image synthesis, and physics-based simulations** to create realistic lighting effects.



Figure 2.16 This close-up of a statue shows both diffuse (smooth shading) and specular (shiny highlight) reflection, as well as darkening in the grooves and creases due to reduced light visibility and interreflections.

To calculate the amount of light exiting a surface point $\mathbf{p}$ in a direction $\hat{\mathbf{v}}_r$ under a given lighting condition, we integrate the product of the incoming light $L_i(\hat{\mathbf{v}}_i; \lambda)$ with the BRDF (some authors call this step a *convolution*). Taking into account the *foreshortening* factor $\cos^+ \theta_i$, we obtain

$$L_r(\hat{\mathbf{v}}_r; \lambda) = \int L_i(\hat{\mathbf{v}}_i; \lambda) f_r(\hat{\mathbf{v}}_i, \hat{\mathbf{v}}_r, \hat{\mathbf{n}}; \lambda) \cos^+ \theta_i d\hat{\mathbf{v}}_i,$$

where

$$\cos^+ \theta_i = \max(0, \cos \theta_i).$$

If the light sources are discrete (a finite number of point light sources), we can replace the integral with a summation,

$$L_r(\hat{\mathbf{v}}_r; \lambda) = \sum_i L_i(\lambda) f_r(\hat{\mathbf{v}}_i, \hat{\mathbf{v}}_r, \hat{\mathbf{n}}; \lambda) \cos^+ \theta_i.$$

BRDFs for a given surface can be obtained through physical modeling heuristic modeling, or through empirical observation. Typical BRDFs can often be split into their diffuse and specular components, as described below.

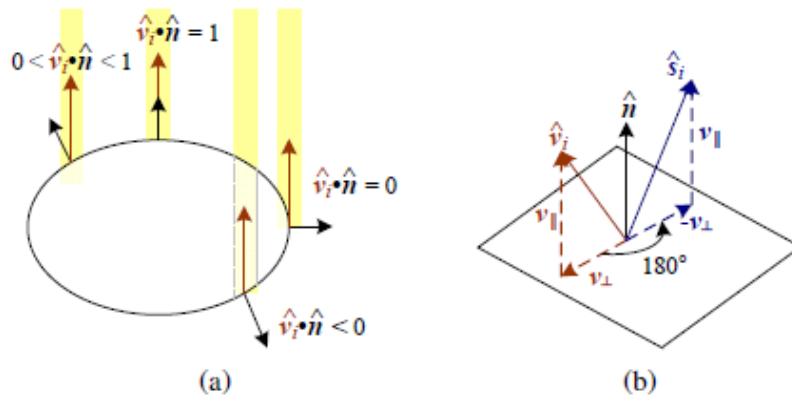


Figure 2.17 (a) The diminution of returned light caused by foreshortening depends on $\hat{\mathbf{v}}_i \cdot \hat{\mathbf{n}}$, the cosine of the angle between the incident light direction $\hat{\mathbf{v}}_i$ and the surface normal $\hat{\mathbf{n}}$. (b) Mirror (specular) reflection: The incident light ray direction $\hat{\mathbf{v}}_i$ is reflected onto the specular direction $\hat{\mathbf{s}}_i$ around the surface normal $\hat{\mathbf{n}}$.

➤ Diffuse Reflection

Diffuse reflection, also known as **Lambertian reflection**, occurs when light is scattered uniformly in all directions upon striking a surface. This type of reflection is responsible for the soft, non-shiny appearance of materials such as painted walls, paper, or statues.

Unlike specular reflection, where light is reflected in a single direction, **diffuse reflection** results from the selective absorption and re-emission of light within the material. This interaction often gives objects their characteristic **body color**, as certain wavelengths are absorbed while others are reflected.

While light is scattered uniformly in all directions, i.e., the BRDF is constant,

$$f_d(\hat{\mathbf{v}}_i, \hat{\mathbf{v}}_r, \hat{\mathbf{n}}; \lambda) = f_d(\lambda),$$

Effect of Incident Angle on Diffuse Reflection

The amount of light reflected from a surface depends on the angle between the **incident light direction** and the **surface normal** (denoted as θ_i). At oblique angles, the same amount of light spreads over a larger surface area, reducing its intensity.

When the surface normal points **away** from the light source, the surface becomes completely **self-shadowed**, receiving little to no illumination. A practical example is how we adjust our body's orientation toward the Sun or a fireplace to maximize warmth. Similarly, a flashlight shining directly on a wall appears brighter than when projected at an angle. The shading equation for **diffuse reflection** models this effect mathematically to determine surface illumination.

$$L_d(\hat{\mathbf{v}}_r; \lambda) = \sum_i L_i(\lambda) f_d(\lambda) \cos^+ \theta_i = \sum_i L_i(\lambda) f_d(\lambda) [\hat{\mathbf{v}}_i \cdot \hat{\mathbf{n}}]^+,$$

➤ Specular Reflection

Specular reflection (gloss or highlight reflection) depends on the outgoing light direction. For a mirrored surface, light reflects in a direction rotated 180° around the surface normal $\hat{\mathbf{n}}$ (Figure 2.17b). This means that incident light rays follow a predictable reflection path, computed using geometric principles. In simpler terms, specular reflection creates sharp highlights on smooth surfaces, like the shine on polished metal or a water surface.

$$\hat{\mathbf{s}}_i = \mathbf{v}_{\parallel} - \mathbf{v}_{\perp} = (2\hat{\mathbf{n}}\hat{\mathbf{n}}^T - \mathbf{I})\mathbf{v}_i.$$

The amount of light reflected in a given direction $\hat{\mathbf{v}}_r$ thus depends on the angle $\theta_s = \cos^{-1}(\hat{\mathbf{v}}_r \cdot \hat{\mathbf{s}}_i)$ between the view direction $\hat{\mathbf{v}}_r$ and the specular direction $\hat{\mathbf{s}}_i$. For example, the Phong (1975) model uses a power of the cosine of the angle,

$$f_s(\theta_s; \lambda) = k_s(\lambda) \cos^{k_e} \theta_s, \quad (2.91)$$

while the Torrance and Sparrow (1967) micro-facet model uses a Gaussian,

$$f_s(\theta_s; \lambda) = k_s(\lambda) \exp(-c_s^2 \theta_s^2). \quad (2.92)$$

Larger exponents k_e (or inverse Gaussian widths c_s) correspond to more specular surfaces with distinct highlights, while smaller exponents better model materials with softer gloss.

➤ Phong Shading

Phong (1975) introduced a shading model that combines **diffuse reflection**, **specular reflection**, and an additional component called **ambient illumination**. This ambient term accounts for general, indirect lighting caused by inter-reflections (such as light bouncing off walls) or distant sources like the sky. Unlike diffuse and specular reflections, the ambient component does not depend on surface orientation but is influenced by the colors of both the **ambient illumination** $L_a(\lambda)$ and the object's **ambient reflectivity** $k_a(\lambda)$.

$$f_a(\lambda) = k_a(\lambda)L_a(\lambda). \quad (2.93)$$

Putting all of these terms together, we arrive at the *Phong shading* model,

$$L_r(\hat{\mathbf{v}}_r; \lambda) = k_a(\lambda)L_a(\lambda) + k_d(\lambda) \sum_i L_i(\lambda)[\hat{\mathbf{v}}_i \cdot \hat{\mathbf{n}}]^+ + k_s(\lambda) \sum_i L_i(\lambda)(\hat{\mathbf{v}}_r \cdot \hat{\mathbf{s}}_i)^{k_e}. \quad (2.94)$$

Computer Vision: Algorithms and Applications, 2nd ed. (final draft, Sept. 2021)

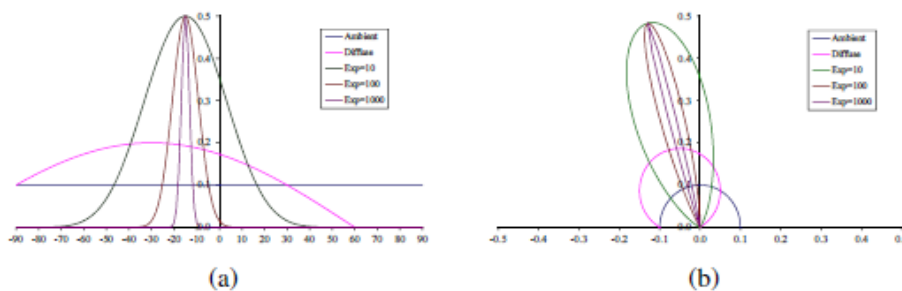


Figure 2.18 Cross-section through a Phong shading model BRDF for a fixed incident illumination direction: (a) component values as a function of angle away from surface normal; (b) polar plot. The value of the Phong exponent k_e is indicated by the “Exp” labels and the light source is at an angle of 30° away from the normal.

Typically, the ambient and diffuse reflection color distributions $k_a(\lambda)$ and $k_d(\lambda)$ are similar since both result from subsurface scattering within the material. The **specular reflection** $k_s(\lambda)$ is usually uniform (white), as interface reflections do not alter the light color. However, metallic materials like copper show color variation in specular reflection.

The ambient illumination $L_a(\lambda)$ often differs in color from direct light sources $L_i(\lambda)$ —for example, it may be **blue** in an outdoor setting or **yellow** in an indoor environment lit by candles or incandescent lights. Shadows appear **bluer** due to ambient sky illumination.

While the **diffuse component** depends on the angle of the incoming light direction $\hat{v}_i$, the **specular component** is influenced by the angle between the viewer $\hat{v}_r$ and the **specular reflection direction** $\hat{s}_i$, which itself depends on $\hat{v}_i$ and the surface normal $\hat{n}$.

Though widely used, the Phong shading model has been replaced by more physically accurate models such as the **Cook-Torrance model** (1982), based on the **microfacet theory** of Torrance and Sparrow (1967). Initially, Phong shading was implemented in hardware, but modern **programmable pixel shaders** allow for more complex and realistic shading models.

➤ Di-chromatic reflection model:

The Torrance and Sparrow (1967) model of reflection also forms the basis of Shafer's (1985) di-chromatic reflection model, which states that the apparent color of a uniform material lit from a single source depends on the sum of two terms,

$$L_r(\hat{v}_r; \lambda) = L_i(\hat{v}_r, \hat{v}_i, \hat{n}; \lambda) + L_b(\hat{v}_r, \hat{v}_i, \hat{n}; \lambda) \quad (2.95)$$

$$= c_i(\lambda)m_i(\hat{v}_r, \hat{v}_i, \hat{n}) + c_b(\lambda)m_b(\hat{v}_r, \hat{v}_i, \hat{n}), \quad (2.96)$$

The reflected light radiance consists of two components: L_i , the radiance reflected at the interface, and L_b , the radiance reflected from the surface body. Each is a product of a **relative power spectrum** $c(\lambda)$ (wavelength-dependent) and a **magnitude function** $m(\hat{v}_r, \hat{v}_i, \hat{n})$ (geometry-dependent). This model, derived from a generalized Phong model, has been effectively used in computer vision for **specular object segmentation** (Klinker, 1993) and has influenced **two-color models** in applications like **Bayer pattern demosaicing** (Bennett et al., 2006).

➤ Global Illumination (Ray Tracing and Radiosity)

Basic shading models assume light travels directly from sources to surfaces and then to the camera. However, in reality, light can be **blocked (shadows)** and **bounce multiple times** before reaching the camera.

Two primary techniques model these effects:

1. **Ray Tracing / Path Tracing** – Ideal for highly **specular scenes** (e.g., glass, mirrors). It traces individual rays, following their reflections and refractions (Glassner, 1995).
2. **Radiosity** – Suited for **diffuse surfaces**, calculating light exchange between surfaces based on form factors and reflectance (Cohen & Wallace, 1993).

Ray Tracing Algorithm:

- Traces rays from the camera to surfaces, computing illumination using shading equations.
- **Shadow maps** can determine if surfaces are directly lit.
- **Secondary rays** simulate reflections, refractions, and light interactions.

Radiosity Algorithm:

- Associates light values with **surface patches** and computes light exchange.
- Uses a linear system to solve for **final illumination** and allows rendering from different viewpoints.
- Does not account for near-field effects like corner darkening, which some **computer vision techniques** have addressed (Nayar et al., 1991).

Both methods influence **scene appearance and 3D interpretation** and can be combined for more accurate **global illumination**.

2.2.3 OPTICS:

After light from a scene enters the camera, it passes through the **lens** before reaching the sensor. A simple **pinhole model** assumes all rays pass through a single projection point. However, to account for **focus, exposure, vignetting, and aberration**, a more advanced **optical model** is needed (Möller, 1988; Ray, 2002; Hecht, 2015).

The **thin lens model** (Figure 2.19) represents a basic lens as a **single glass element** with equal curvature on both sides. According to the **lens equation**, the relationship between the object distance (z_o) and the image distance (z_i) determines where a focused image forms.

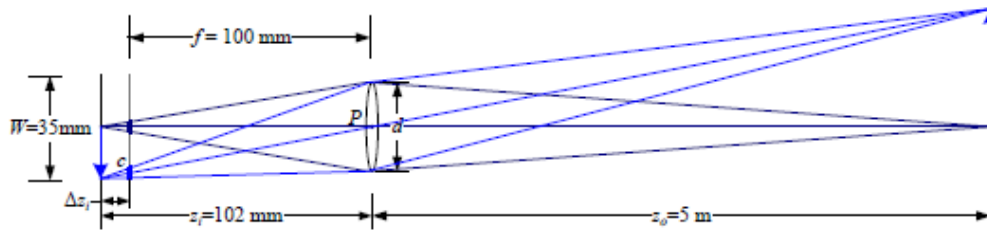


Figure 2.19 A thin lens of focal length f focuses the light from a plane at a distance z_o in front of the lens onto a plane at a distance z_i behind the lens, where $\frac{1}{z_o} + \frac{1}{z_i} = \frac{1}{f}$. If the focal plane (vertical gray line next to c) is moved forward, the images are no longer in focus and the circle of confusion c (small thick line segments) depends on the distance of the image plane motion Δz_i relative to the lens aperture diameter d . The field of view (f.o.v.) depends on the ratio between the sensor width W and the focal length f (or, more precisely, the focusing distance z_i , which is usually quite close to f).

relationship between the distance to an object z_o and the distance behind the lens at which a focused image is formed z_i can be expressed as

$$\frac{1}{z_o} + \frac{1}{z_i} = \frac{1}{f}, \quad (2.97)$$

- Focal Length and Depth of Field:**

The **focal length** (f) of a lens determines where light converges to form a sharp image. When an object is at **infinity** ($z_o \rightarrow \infty$), the image forms at $z_i = f$, making the lens function similarly to a **pinhole camera** at distance f from the focal plane (Figure 2.10).

If the **focal plane** is shifted (e.g., by adjusting the focus ring), objects at z_o become **out of focus**, leading to a **circle of confusion** (c), which depends on the **focus shift** (Δz_i) and **aperture diameter** (d). The range of distances where objects remain acceptably sharp is the **depth of field**, influenced by **focus distance** and **aperture size** (Figure 2.20). The **f-number** ($f/\#$ or N) defines the aperture relative to focal length.

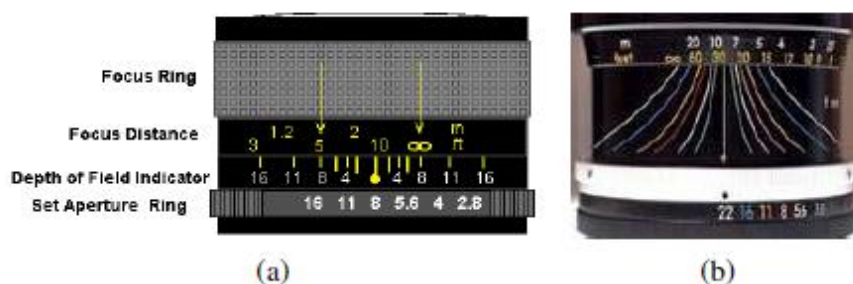


Figure 2.20 Regular and zoom lens depth of field indicators.

Since this depth of field depends on the aperture diameter d , we also have to know how this varies with

the commonly displayed f-number, which is usually denoted as $f/\#$ or N and is defined as

$$f/\# = N = \frac{f}{d}, \quad (2.98)$$

where the focal length f and the aperture diameter d are measured in the same unit (say, millimetres).

Understanding f-Numbers and Aperture

The **f-number** ($f/\#$ or N) represents the ratio of a lens's **focal length** (f) to its **aperture diameter** (d). It is commonly written as $f/1.4$, $f/2$, $f/2.8$, ..., $f/22$, where dividing f by the f-number gives the actual aperture diameter. Alternatively, the notation $N = 1.4, 2, 2.8$, etc. is also used.

- **Full Stops and Exposure Control**

The standard progression of **f-numbers** follows a sequence based on multiples of $\sqrt{2}$ (e.g., 1.4, 2, 2.8, 4, 5.6, etc.). Each step, known as a **full stop**, **halves or doubles** the amount of light entering the lens. This corresponds to a **one-stop change in exposure value (EV)**, which has the same effect on light intensity as doubling or halving the **shutter speed** (e.g., from 1/250s to 1/125s).

- **Depth of Field and Optical Calculations**

By understanding **f-numbers**, one can compute the **depth of field (DoF)**, which depends on the **focal length** (f), **circle of confusion** (c), and **focus distance** (z_0). These relationships can be used to generate **DoF plots**, helping photographers and optics engineers analyze focus characteristics, as demonstrated in **Exercise 2.4**.

Real-World Lens Imperfections

Unlike the **ideal thin lens model**, real lenses suffer from **geometric aberrations** due to their thickness and optical design. **Compound lens elements** help correct these distortions. The **five Seidel aberrations** in third-order optics include:

1. **Spherical Aberration** – Light rays at different distances from the optical axis focus at varying points.
2. **Coma** – Off-axis points appear distorted or elongated.
3. **Astigmatism** – Light rays focusing in different planes produce blurred images.
4. **Curvature of Field** – A flat object appears distorted due to a curved focal plane.
5. **Distortion** – Lines appear bent due to magnification variations across the image.

Understanding these optical principles enables better lens design and image quality optimization (Möller 1988; Ray 2002; Hecht 2015).

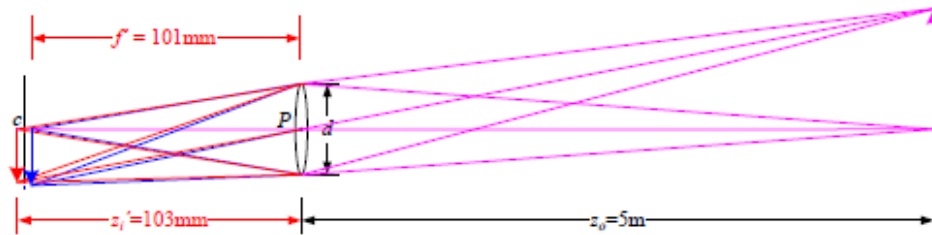


Figure 2.21 In a lens subject to chromatic aberration, light at different wavelengths (e.g., the red and blue arrows) is focused with a different focal length f' and hence a different depth z'_i , resulting in both a geometric (in-plane) displacement and a loss of focus.

➤ Chromatic Aberration in Optical Systems:

1. Causes of Chromatic Aberration

Chromatic aberration arises because the **index of refraction** of glass varies with **wavelength**. This variation causes different colors of light to **focus at slightly different distances** from the lens, resulting in color fringing and blurring in images. The two primary types of chromatic aberration are:

- **Transverse Chromatic Aberration (TCA):** This occurs when different wavelengths experience **different magnification factors**, leading to color shifts at the edges of the image. It can be modeled as **per-color radial distortion** and corrected using **calibration techniques** (see Section 11.1.4).
- **Longitudinal Chromatic Aberration (LCA):** This happens when different wavelengths focus at **different depths** along the optical axis, causing **color-dependent blurring**. While some correction techniques exist (Section 10.1.4), severe longitudinal aberration can cause high-frequency image details to be **irreversibly lost**.

2. Correcting Chromatic Aberration with Compound Lenses

To minimize chromatic and other optical aberrations, **modern photographic lenses** use **compound lens designs** composed of multiple glass elements, each with specialized coatings. Unlike **simple lenses**, compound lenses cannot be approximated by a **single nodal point (P)** in a pinhole model. Instead, they have:

- **Front Nodal Point:** The point where incoming light rays appear to converge before entering the lens.
- **Rear Nodal Point:** The point where light rays exit towards the sensor.

For **camera calibration**, especially in applications like **parallax-free panorama stitching**, determining the **exact location of the front nodal point** is crucial (Section 8.2.3; see Littlefield 2006, Houghton 2013).

3. Challenges in Modeling Wide-Angle and Fisheye Lenses

Not all lenses fit the simple nodal point model. Certain optical systems, including:

- **Ultra-wide-angle (fisheye) lenses**
- **Catadioptric imaging systems** (which combine lenses with curved mirrors, e.g., **Baker and Nayar 1999**)

...do not allow all captured rays to pass through a single nodal point. Instead of using a **traditional lens model**, a **mapping function** or **look-up table** is created to directly relate **pixel coordinates** to corresponding **3D rays in space**. Various research efforts have developed methods for this approach (**Gremban et al. 1988; Champleboux et al. 1992; Grossberg & Nayar 2001; Sturm & Ramalingam 2004; Tardif et al. 2009**), as discussed in **Section 2.1.5**.

By understanding and compensating for chromatic aberration and lens distortions, photographers and optical engineers can significantly enhance image quality and camera performance.

➤ **Vignetting in Camera Lenses**

1. What is Vignetting?

Vignetting is a phenomenon in which the brightness of an image decreases towards the edges. This happens due to the way light interacts with the lens and sensor.

There are two main types of vignetting:

1. **Natural Vignetting** – Caused by the way light travels through the lens.
2. **Mechanical Vignetting** – Caused by physical obstructions inside the lens.

2. Natural Vignetting (Due to Light Behavior)

Natural vignetting happens because light from objects at an angle (β) reaches the lens with reduced intensity. The reasons for this include:

- **Foreshortening of the Object Surface:**
 - When an object is viewed at an angle, it appears smaller to the lens.
 - This reduces the amount of light captured by the camera.
 - Light intensity drops by a factor of $\cos^4(\beta)$.

- Distance-Based Light Fall-off (Inverse-Square Law):
 - Light intensity decreases with distance ($1/r^2$ rule).
 - If an object is far, its light spreads out more, reducing brightness.
- Foreshortening of the Lens Aperture:
 - The lens opening appears smaller at an angle.
 - Instead of a full circle, the aperture looks elliptical.
 - This further reduces the amount of light reaching the sensor.

Together, these effects cause brightness to drop more at the edges of the image than at the center.

Mathematical Formula for Natural Vignetting:

The light reaching the camera sensor E is related to:

$$E = L \times 4 \left(\frac{d}{f} \right)^2 \cos^4 \beta$$

- Scene radiance (L) – The amount of light from the object.
- Aperture diameter (d) – The size of the lens opening.
- Focal length (f) – The distance from the lens to the focus point.
- Angle (β) – The off-centre angle at which light enters.

This formula shows that:

- Larger aperture (d) → More light enters.
- Greater focal length (f) → Less light reaches the sensor.
- Larger angle (β) → More vignetting occurs due to $\cos^4(\beta)$ fall-off.

3. Mechanical Vignetting (Due to Lens Design)

Unlike natural vignetting, mechanical vignetting happens when parts of the lens block light near the edges.

This can be caused by:

- Lens housing or internal elements blocking peripheral rays.
- Using thick filters in front of the lens.

Unlike natural vignetting, mechanical vignetting can be reduced by using a smaller aperture (higher f-number).

Two kinds of phenomena usually contribute to this effect (Ray 2002). The first is called *natural vignetting* and is due to the foreshortening in the object surface, projected pixel, and lens aperture, as shown in Figure 2.22. Consider the light leaving the object surface patch of size δo located at an *off-axis angle* α . Because this patch is foreshortened with respect to the camera lens, the amount of light reaching the lens is reduced by a factor $\cos \alpha$. The amount of light reaching the lens is also subject to the usual $1/r^2$ fall-off; in this case, the distance $r_o = z_o / \cos \alpha$. The actual area of the aperture through which the light passes is foreshortened by an additional factor $\cos \alpha$, i.e., the aperture as seen from point O is an ellipse of dimensions $d \times d \cos \alpha$. Putting all of these factors together, we see that the amount of light leaving O and passing through the aperture on its way to the image pixel located at I is proportional to

$$\frac{\delta o \cos \alpha}{r_o^2} \pi \left(\frac{d}{2} \right)^2 \cos \alpha = \delta o \frac{\pi d^2}{4 z_o^2} \cos^4 \alpha. \quad (2.99)$$

Since triangles $\triangle OPQ$ and $\triangle IPJ$ are similar, the projected areas of the object surface δo and image pixel δi are in the same (squared) ratio as $z_o : z_i$,

$$\frac{\delta o}{\delta i} = \frac{z_o^2}{z_i^2}. \quad (2.100)$$

2.3 The digital camera

Putting these together, we obtain the final relationship between the amount of light reaching pixel i and the aperture diameter d , the focusing distance $z_i \approx f$, and the off-axis angle α ,

$$\delta o \frac{\pi d^2}{4 z_o^2} \cos^4 \alpha = \delta i \frac{\pi d^2}{4 z_i^2} \cos^4 \alpha \approx \delta i \frac{\pi}{4} \left(\frac{d}{f} \right)^2 \cos^4 \alpha, \quad (2.101)$$

4. How to Reduce Vignetting?

Vignetting is **not always a bad thing**—some photographers use it **artistically** to draw attention to the center of an image. However, if unwanted, here's how to reduce it:

- ✓ Use a **smaller aperture (higher f-number)** to allow light from wider angles.
- ✓ Choose **high-quality lenses** designed to minimize vignetting.
- ✓ Apply **post-processing corrections** in software like Adobe Lightroom or Photoshop.
- ✓ Avoid **thick lens filters** that may block peripheral light.

By understanding vignetting, photographers and students can **improve their image composition** and optimize camera settings for better brightness distribution across the frame.

2.3 Digital Camera

When light travels from a source, reflects off surfaces, and passes through a camera's lens, it eventually reaches the imaging sensor. The question is: how does this light get converted into the digital RGB (Red, Green, Blue) values that make up a digital image?

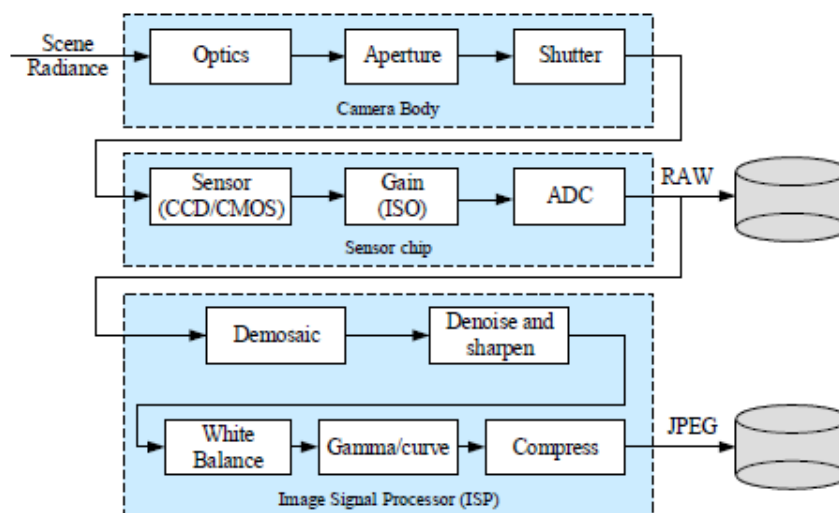


Figure 2.23 Image sensing pipeline, showing the various sources of noise as well as typical digital post-processing steps.

A digital camera follows a series of steps to process light into an image. This involves exposure settings (shutter speed and gain), non-linear adjustments, sampling, aliasing, and noise reduction. Researchers like Healey and Kondepudy (1994), Tsing, Ramesh, and Kanade (2001), and Liu, Szeliski et al. (2008) have developed simplified models to explain these processes. More advanced models, such as those by

Chakrabarti, Scharstein, and Zickler (2009), use 24 parameters for better accuracy. Recent research by Brooks, Mildenhall et al. (2019) has focused on reversing the in-camera processing of JPEG images to restore their original RAW format, which helps in reducing noise.

When light reaches the camera sensor, it is detected by an active sensing area. The sensor integrates light for a specific duration, known as exposure time (or shutter speed), measured in fractions of a second like $1/125$, $1/60$, or $1/30$. The captured signal is then passed through amplifiers before being converted into digital data.

There are two main types of image sensors used in modern digital cameras:

1. Charge-Coupled Device (CCD)

- In CCD sensors, light is stored in tiny wells (pixels) during exposure.
- The accumulated charge is transferred in a step-by-step manner (like a bucket brigade) to sense amplifiers, which convert the signal for digital processing.
- Older CCDs had an issue called blooming, where excess charge from overexposed pixels spilled into neighboring ones. However, modern CCDs have anti-blooming technology to prevent this.

2. Complementary Metal-Oxide-Semiconductor (CMOS)

- In CMOS sensors, photons directly influence the conductivity of electronic components.
- Each pixel has its own amplifier, making data transfer faster and more power-efficient than CCDs.
- CMOS sensors are widely used in modern digital cameras, smartphones, and video recording devices.

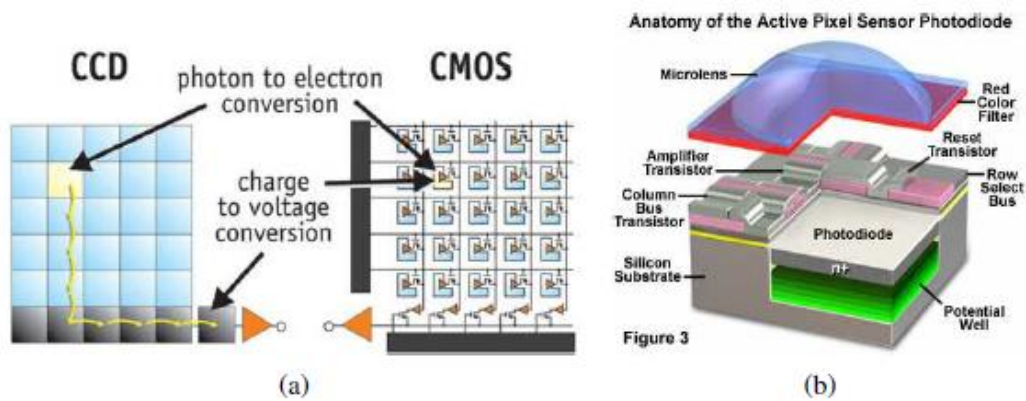


Figure 2.24 Digital imaging sensors: (a) CCDs move photogenerated charge from pixel to pixel and convert it to voltage at the output node; CMOS imagers convert charge to voltage inside each pixel (Litwiller 2005) © 2005 Photonics Spectra; (b) cutaway diagram of a CMOS pixel sensor, from <https://micro.magnet.fsu.edu/primer/digitalimaging/cmosimagesensors.html>.

Key Factors Affecting Image Sensor Performance

- **Shutter**

Speed:

Controls how long light is allowed to hit the sensor.

- **Fast shutter speed (e.g., 1/1000s):** Reduces motion blur, useful for action shots.
- **Slow shutter speed (e.g., 1/30s):** Allows more light in, useful in low-light conditions.

- **Sampling Pitch:**

The physical spacing between sensor pixels.

- Smaller pitch = higher resolution but less light sensitivity.
- Larger pitch = better light capture, reducing noise.

- **Fill Factor:**

The percentage of the sensor's surface that actively captures light.

- Higher fill factor = better light capture, less aliasing.
- Modern sensors improve this using *backside illumination* and microlenses.

- **Chip Size:**

- Smaller sensors (used in compact cameras) limit light capture.
- Larger sensors (used in DSLRs) capture more light, improving image quality.

- **Analog Gain & ISO:**

- Amplifies the signal before converting it to digital.
- Higher ISO (e.g., 800, 1600) helps in low-light but increases noise.

- **Sensor Noise:**

- Comes from different sources (e.g., sensor defects, electrical interference, photon limitations).
 - Noise is more noticeable in low-light images.
 - Cameras use noise reduction algorithms to improve quality.
 - **ADC (Analog-to-Digital Conversion):**
 - Converts light into digital values (e.g., 8-bit JPEG or 16-bit RAW).
 - More bits = better dynamic range, but actual usable bits depend on noise levels.
-

Digital Image Processing

Once the sensor captures the raw image, digital processing enhances it before saving the final file. Key processes include:

- **Demosaicing:** Converts raw sensor data into a full-color image.
 - **White Balance:** Adjusts color to match real-world lighting conditions.
 - **Gamma Correction:** Adjusts brightness and contrast for better display.
 - **Compression:** Reduces file size (e.g., JPEG) while preserving quality.
-

Advancements in Camera Technology

Modern sensors are improving rapidly with advancements in **image processing, noise reduction, and depth sensing**. Conferences like the *IS&T Symposium on Electronic Imaging Science and Technology* and sources like the *Image Sensors World blog* track these developments.

2.3.1 Sampling and aliasing

- Light falling on an image sensor is captured, integrated, and digitized.
- If the **fill factor** of the sensor is small and the signal is **not band-limited**, aliasing occurs.

Aliasing Phenomenon

- Consider a **1D signal** with two sine waves:
 - **Frequency $f = 3/4$**
 - **Frequency $f = 5/4$**
- If sampled at **$f = 2$** , both waves produce identical sampled values.
- This leads to **aliasing**, where we cannot determine the original frequency.

Shannon's Sampling Theorem

- States that a signal must be sampled at a rate **at least twice** its highest frequency to avoid aliasing.

- If the sampling rate is too low, different signals become **indistinguishable**.

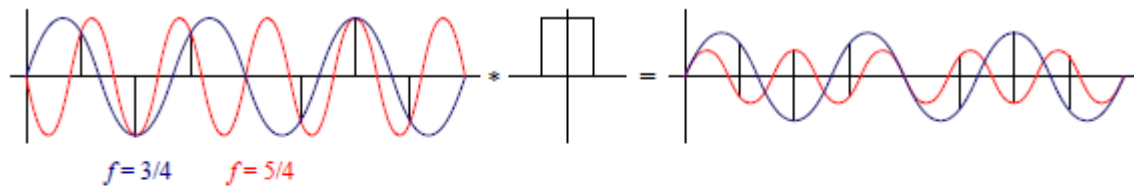


Figure 2.25 Aliasing of a one-dimensional signal: The blue sine wave at $f = 3/4$ and the red sine wave at $f = 5/4$ have the same digital samples, when sampled at $f = 2$. Even after convolution with a 100% fill factor box filter, the two signals, while no longer of the same magnitude, are still aliased in the sense that the sampled red signal looks like an inverted lower magnitude version of the blue signal. (The image on the right is scaled up for better visibility. The actual sine magnitudes are 30% and -18% of their original values.)

from its instantaneous samples must be at least twice the highest frequency,¹⁹

$$f_s \geq 2f_{\max}. \quad (2.103)$$

The maximum frequency in a signal is known as the *Nyquist frequency* and the inverse of the minimum sampling frequency $r_s = 1/f_s$ is known as the *Nyquist rate*.

Aliasing and Downsampling: Effect of Averaging in Imaging Chips

- Imaging chips **average** the light field over a finite area, attenuating higher frequencies.
- However, even with a **100% fill factor**, frequencies above the **Nyquist limit** still cause aliasing, though at a reduced magnitude.

Downsampling and Aliasing

- Aliasing is more noticeable when **downsampling** with a poor-quality filter (e.g., **box filter**).
- Example in **Figure 2.26**:
 - **High-frequency chirp image** (where frequencies increase over time).
 - Results of sampling with:
 - **25% fill-factor area sensor**

- **100% fill-factor sensor**
- **High-quality 9-tap filter**
- More examples of **decimation filters** in Section 3.5.2 and Figure 3.29.

Point Spread Function (PSF) & Aliasing Prediction

- **PSF (Point Spread Function)** helps estimate aliasing effects.
- Represents the **sensor's response** to an ideal **point light source**.
- PSF is a result of:
 - **Optical system (lens) blur**
 - **Finite sensor resolution**

Point Spread Function (PSF) and Aliasing

Understanding the PSF

- The Point Spread Function (PSF) is derived by convolving:
 1. Blur function of the lens
 2. Fill factor (sensor area shape and spacing)
 3. Response of the anti-aliasing filter (if present)
- Figure 2.27a:
 - Shows a 1D cross-section of a PSF for a lens with a disc blur.
 - The blur has a radius equal to the pixel spacing (g).
 - The sensing chip has a horizontal fill factor of 80%.

Modulation Transfer Function (MTF) & Aliasing

- Fourier transform of the PSF gives the MTF (Modulation Transfer Function).

- The MTF estimates aliasing based on the area of the Fourier magnitude outside the Nyquist frequency ($f \leq f_s$).
- Defocusing the lens increases the blur radius to $2s$ (Figure 2.27c):
 - Aliasing decreases significantly.
 - But image detail also decreases (frequencies closer to $f = f_s$ are lost).

Estimating PSF in Laboratory Conditions

- Using a pinhole:
 - A point light source is placed behind a black pinhole in a cardboard piece.
 - The pinhole's image forms the PSF, but only at pixel-level accuracy.
 - It models large blur (e.g., defocus blur) but not sub-pixel detail.
- Alternative method (Section 10.1.4):
 - Uses a calibration pattern (e.g., slanted step edges).

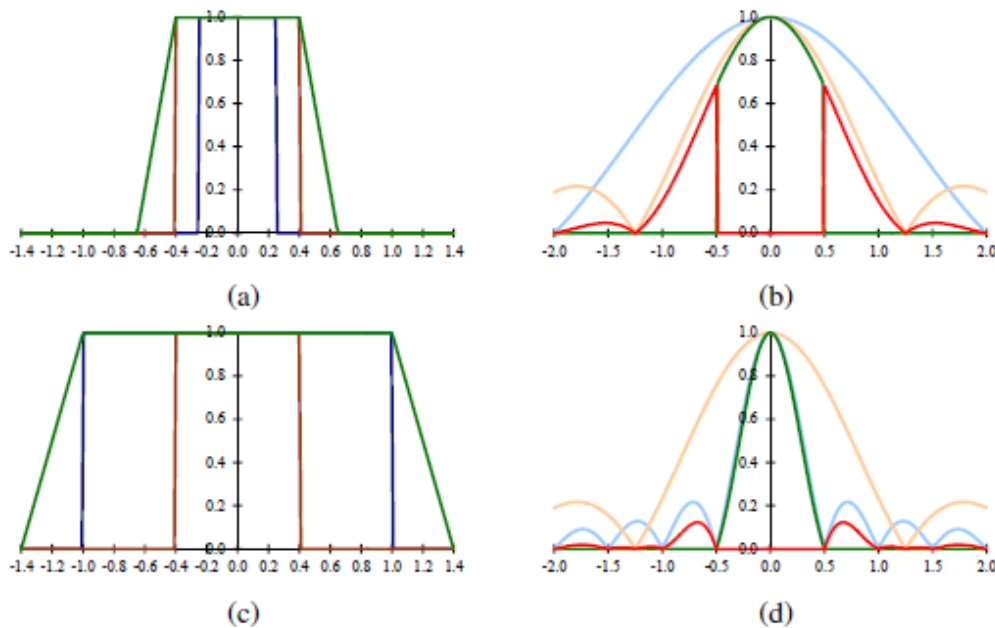


Figure 2.27 Sample point spread functions (PSF): The diameter of the blur disc (blue) in (a) is equal to half the pixel spacing, while the diameter in (c) is twice the pixel spacing. The horizontal fill factor of the sensing chip is 80% and is shown in brown. The convolution of these two kernels gives the point spread function, shown in green. The Fourier response of the PSF (the MTF) is plotted in (b) and (d). The area above the Nyquist frequency where aliasing occurs is shown in red.

2.3.2 Color Theory: Additive & Subtractive Colors

Subtractive Color Mixing

- The **traditional** color mixing idea (e.g., **blue + yellow = green**) is **incorrect**.

- The **correct subtractive primary colors** are:
 - Cyan (light blue green)
 - Magenta (pink)
 - Yellow
- **Black** is also used in **four-color printing (CMYK)**.
- **Subtractive colors absorb certain wavelengths** in the color spectrum.

Additive Color Mixing

- **Primary colors in additive mixing:**
 - Red, Green, Blue (RGB)
- These combine (e.g., in TVs & monitors) to produce:
 - Cyan, Magenta, Yellow, White, etc.
- **Additive colors are used in light-emitting devices.**

How Do Two Colors Create a Third Color?

- When red and green produce yellow, are wavelengths being mixed?
- No, colors do not mix at a wavelength level.
- This effect is due to the tri-stimulus (or tri-chromatic) nature of human vision:
 - The eye has three types of cone cells, each responding to different wavelengths.
 - Our brain interprets combined signals as new colors.

Note that for machine vision applications, such as remote sensing and terrain classification, it is preferable to use many more wavelengths. Similarly, surveillance applications can often benefit from sensing in the near-infrared (NIR) range.

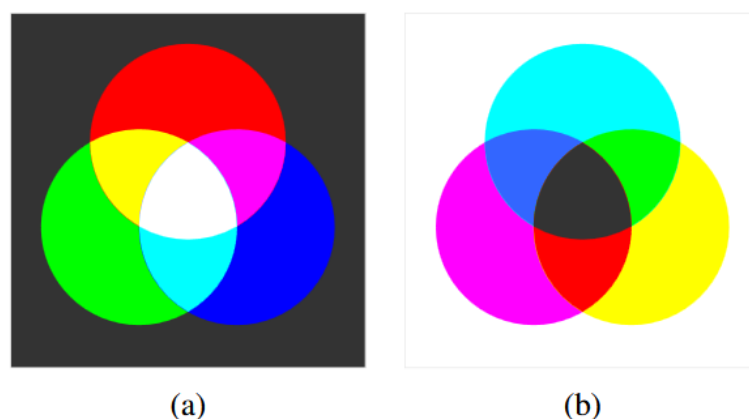


Figure 2.28 *Primary and secondary colors: (a) additive colors red, green, and blue can be mixed to produce cyan, magenta, yellow, and white; (b) subtractive colors cyan, magenta, and yellow can be mixed to produce red, green, blue, and black.*

➤ CIE RGB and XYZ :

Based on the **tri-chromatic theory of perception**, which suggests that all **monochromatic** (single-wavelength) colors can be reproduced using a mix of **three primary colors**.

- The CIE (Commission Internationale d'Éclairage) standardized the **RGB representation** in the **1930s**.
- **Primary colors and their wavelengths:**
 - **Red** → 700.0 nm
 - **Green** → 546.1 nm
 - **Blue** → 435.8 nm

1. Color Matching Experiment

- Performed with a **standard observer** (average perceptual results over multiple subjects).
- **Key findings:**
 - For certain blue-green spectra, **negative amounts of red had to be added** to achieve a color match.
 - This means that **some colors outside the RGB spectrum cannot be accurately represented using positive RGB values**.

2. Metamers & Lighting Effects

- **Metamers:** Different colors with different spectral compositions that **appear identical under certain lighting**.
- **Example:** Two fabrics or paints may match in one lighting condition but look different in another.

3. XYZ Color Space

- Developed by the CIE to address issues in the **RGB model**, especially the problem of mixing **negative light**.
- **XYZ contains all pure spectral colors with only positive values**, making it more practical for color representation.

➤ CIE XYZ Color Space

RGB to XYZ Transformation

- The transformation from RGB to XYZ is given by the matrix equation:

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \frac{1}{0.17697} \begin{bmatrix} 0.49 & 0.31 & 0.20 \\ 0.17697 & 0.81240 & 0.01063 \\ 0.00 & 0.01 & 0.99 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

- This transformation normalizes the Y value so that **pure red corresponds to a Y value of 1**.
- The **Y axis in XYZ represents luminance** (perceived brightness).
- The RGB triplet (1,1,1) maps to a Y value of 1, ensuring that white remains balanced.

Chromaticity Coordinates

- By dividing XYZ values by their sum ($X + Y + Z$), we obtain the **chromaticity coordinates**:

$$x = \frac{X}{X + Y + Z}, \quad y = \frac{Y}{X + Y + Z}, \quad z = \frac{Z}{X + Y + Z}$$

- These coordinates **discard absolute intensity** and retain only **pure color information**.
- The **sum of x, y, and z is always 1**.

Chromaticity Diagram

- The chromaticity diagram is formed by sweeping the monochromatic wavelength λ from 380 nm to 800 nm.
- Key observations:
 - The outer curve represents all possible spectral colors.
 - The straight-line segment (purple line) connects the endpoints and represents non-spectral purples.
 - The inset triangle defines red, green, and blue, aligning with RGB color matching experiments.

L*a*b_ color space

While the XYZ color space has many convenient properties, including the ability to separate luminance from chrominance, it does not actually predict how well humans perceive differences in color or luminance.

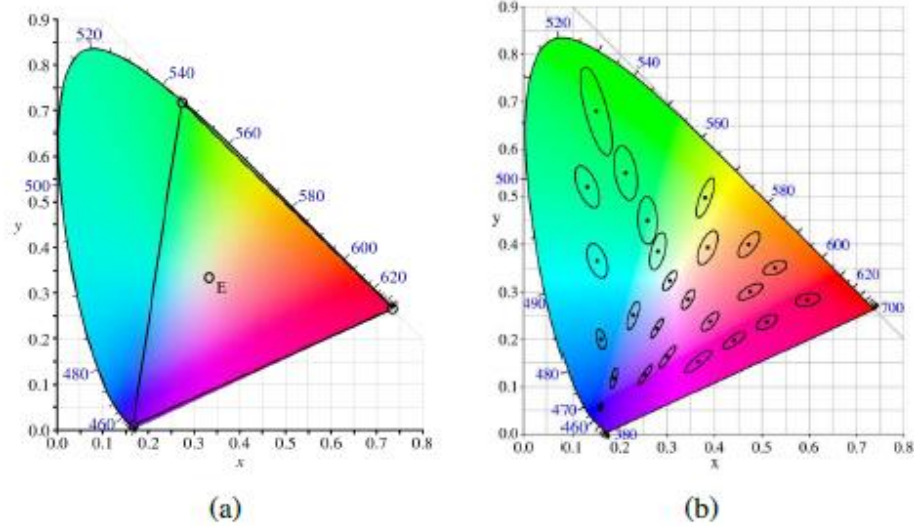


Figure 2.30 CIE chromaticity diagram, showing the pure single-wavelength spectral colors along the perimeter and the white point at E, plotted along their corresponding (x, y) values. (a) the red, green, and blue primaries do not span the complete gamut, so that negative amounts of red need to be added to span the blue–green range; (b) the MacAdam ellipses show color regions of equal discriminability, and form the basis of the Lab perceptual color space.

Because the response of the human visual system is roughly logarithmic (we can perceive *relative* luminance differences of about 1%), the CIE defined a non-linear re-mapping of the XYZ space called $L^*a^*b^*$ (also sometimes called CIELAB), where differences in luminance or chrominance are more perceptually uniform, as shown in Figure 2.30b.²⁵

The L^* component of *lightness* is defined as

$$L^* = 116f\left(\frac{Y}{Y_n}\right), \quad (2.106)$$

where Y_n is the luminance value for nominal white (Fairchild 2013) and

$$f(t) = \begin{cases} t^{1/3} & t > \delta^3 \\ t/(3\delta^2) + 2\delta/3 & \text{else,} \end{cases} \quad (2.107)$$

is a finite-slope approximation to the cube root with $\delta = 6/29$. The resulting 0...100 scale roughly measures equal amounts of lightness perceptibility.

In a similar fashion, the a^* and b^* components are defined as

$$a^* = 500 \left[f\left(\frac{X}{X_n}\right) - f\left(\frac{Y}{Y_n}\right) \right] \quad \text{and} \quad b^* = 200 \left[f\left(\frac{Y}{Y_n}\right) - f\left(\frac{Z}{Z_n}\right) \right], \quad (2.108)$$

where again, (X_n, Y_n, Z_n) is the measured white point. Figure 2.33i–k show the $L^*a^*b^*$ representation for a sample color image.

➤ Color cameras

RGB Video Cameras and Color Representation

How RGB Cameras Work

- RGB and XYZ models describe perceived color but **do not explain how cameras capture color**.
- Color monitors do **not measure only the nominal red (700 nm), green (546.1 nm), and blue (435.8 nm) wavelengths**.
- Monitors mix different wavelengths to reproduce colors, such as emitting **negative red light** for cyan reproduction.

Standard Definition & HDTV Color Models

- **Early RGB video cameras** relied on **phosphors** in CRT displays.
- The **NTSC standard (ITU-R BT.601)** mapped RGB values to **XYZ values** for consistent color perception.
- HDTV and modern displays follow **ITU-R BT.709**, which defines the transformation from **RGB to XYZ**:

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} 0.412453 & 0.357580 & 0.180423 \\ 0.212671 & 0.715160 & 0.072169 \\ 0.019334 & 0.119193 & 0.950227 \end{bmatrix} \begin{bmatrix} R_{709} \\ G_{709} \\ B_{709} \end{bmatrix}$$

Spectral Response in Cameras

- Cameras **do not capture discrete RGB values**; instead, they integrate light across a **range of wavelengths**:

$$R = \int L(\lambda) S_R(\lambda) d\lambda$$

$$G = \int L(\lambda) S_G(\lambda) d\lambda$$

$$B = \int L(\lambda) S_B(\lambda) d\lambda$$

- Here, $L(\lambda)$ is the **light spectrum**, and $S_R(\lambda)$, $S_G(\lambda)$, $S_B(\lambda)$ are the **sensor response functions** for red, green, and blue.
- Color monitors **do not strictly emit single wavelengths**; they create colors through **spectral mixing**.
- Different **display technologies** (CRT, HDTV, modern LCD) use different **RGB-to-XYZ mappings**.
- Cameras **integrate color over a spectrum**, not just at fixed wavelengths.

➤ **Camera Spectral Sensitivities and Color Filter Arrays:**

Camera spectral sensitivities are not always publicly available unless the manufacturer provides data or they are measured using monochromatic light. Standards like BT.709 do not define these sensitivities, only the transformation needed to match standard color values. This means manufacturers can use different sensor sensitivities as long as they can be converted to standard RGB values through a linear transformation.

TV and computer monitors follow standard RGB output rules but can apply digital transformations before displaying colors. Properly calibrated monitors provide color management, ensuring consistency between real-life colors, screens, and printed images.

Early color cameras used three separate tubes or RGB sensors for capturing images. Modern digital cameras, however, use a **Color Filter Array (CFA)**, where individual sensors are covered with color filters instead of using separate chips for red, green, and blue.

The **Bayer pattern** is the most common CFA used today. It has twice as many green filters as red and blue, arranged in a checkerboard-like structure. This is because human vision is more sensitive to luminance (brightness) than to chrominance (color). Green light plays a major role in luminance, which helps maintain sharpness in images. This property is also used in color image compression techniques.

Since each pixel only captures one color, missing color values are estimated using a process called **demosaicing**. This technique reconstructs full-color images by filling in the missing red, green, and blue values for each pixel.

Similarly, color **LCD monitors** use alternating stripes of red, green, and blue filters in front of liquid crystal areas. This setup creates the perception of full-color images on the screen.

The human visual system is more sensitive to brightness (luminance) than to color details (chrominance). Because of this, digital techniques can be used to enhance the sharpness of images. By applying special filtering methods to RGB or monochrome images, the perception of crispness can be improved, making images appear clearer and more detailed.

➤ **Color Balance:**

Cameras adjust color balance before encoding RGB values to ensure that white areas appear truly white. This process shifts the white point of an image to compensate for different lighting conditions. If the camera and lighting system match (such as using

daylight illuminant D65 in BT.709), minimal adjustment is needed. However, under strong-colored lighting, like warm indoor lighting that adds a yellow or orange tint, significant corrections are required.

A simple way to adjust color balance is by multiplying each RGB value by a specific factor, effectively scaling the colors to achieve a more neutral white. More advanced methods use a 3×3 color transformation matrix, which can map colors to XYZ space and back, creating a more refined correction known as a "color twist."

➤ Gamma

Early black-and-white televisions used phosphors in CRT displays that responded non-linearly to voltage, meaning the brightness did not increase proportionally with the input signal. This relationship is described by a value called gamma (γ), which characterizes how the display converts input voltage into visible brightness.

$$B = V^\gamma,$$

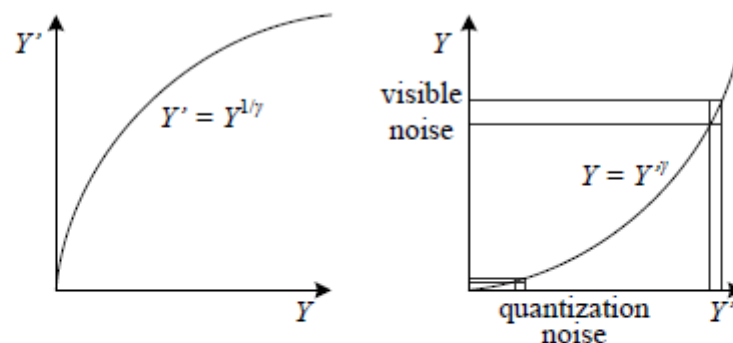


Figure 2.32 Gamma compression: (a) The relationship between the input signal luminance Y and the transmitted signal Y' is given by $Y' = Y^{1/\gamma}$. (b) At the receiver, the signal Y' is exponentiated by the factor γ , $\hat{Y} = Y'^\gamma$. Noise introduced during transmission is squashed in the dark regions, which corresponds to the more noise-sensitive region of the visual system.

with a γ of about 2.2. To compensate for this effect, the electronics in the TV camera would pre-map the sensed luminance Y through an inverse gamma,

$$Y' = Y^{\frac{1}{\gamma}}, \quad (2.112)$$

with a typical value of $\frac{1}{\gamma} = 0.45$.

Gamma Correction in Imaging and Its Challenges

In the early days of analog television, signals were adjusted using gamma correction before being transmitted. This adjustment had an unexpected benefit—it reduced noise in darker areas of the image, making them appear clearer. This worked well because human vision is more sensitive to changes in brightness rather than absolute brightness levels.

When color television was introduced, the same gamma correction process was applied separately to red, green, and blue signals before they were combined and encoded. Even though modern digital systems no longer have analog noise issues, gamma correction is still useful because digital signals are compressed, and applying inverse gamma helps maintain image quality.

However, gamma correction can be problematic in computer vision and graphics. Many image-processing techniques, such as shading and lighting simulations, assume a linear brightness scale. For accurate results, calculations should be performed in a linear space before gamma correction is applied for display. Unfortunately, many graphics systems do not follow this approach, leading to errors in brightness and color representation.

Gamma Correction and Its Role in Computer Vision

Many computer graphics systems directly use RGB values and display them without adjusting for gamma. However, newer imaging standards like 16-bit sRGB use a linear space, reducing gamma-related issues.

In computer vision, working with accurate brightness levels is essential. Techniques like **photometric stereo** (used to estimate surface details) and **image deblurring** require brightness values to be in a linear intensity scale. Therefore, before performing precise image calculations, it's important to **remove gamma correction** and undo any automatic color adjustments made by the camera.

Researchers like Chakrabarti, Scharstein, and Zickler (2009) developed a 24-parameter model to accurately simulate modern digital camera processing. They also provide a dataset of images useful for testing.

However, in some vision applications, such as **feature detection**, **stereo vision**, or **motion estimation**, linearization is not always necessary. Deciding whether to remove gamma requires careful thought—for example, in **image stitching**, adjusting for brightness differences might not require full gamma correction.

Understanding these processing steps can be complex. A useful way to explore these effects is through hands-on testing—taking pictures of color charts and comparing RAW images with JPEG-compressed versions.

➤ Color Spaces

RGB and XYZ are the main color spaces used to describe color signals. However, other representations have been developed for video and image processing.

One of the earliest color representations for video transmission was the **YIQ standard** (used in NTSC video in North America) and the **YUV standard** (used in PAL video in Europe). These systems were designed to include a **luma (Y) channel**, which mimics true luminance and is similar to a black-and-white TV signal, along with two **chroma channels** for color information.

In these systems, the **Y' (luma) signal** is computed as follows:

The UV components are derived from scaled versions of $(B' - Y')$ and $(R' - Y')$, namely,

$$U = 0.492111(B' - Y') \quad \text{and} \quad V = 0.877283(R' - Y'), \quad (2.115)$$

For NTSC (BT.601 standard):

$$Y'_{601} = 0.299R' + 0.587G' + 0.114B'$$

Here, $R'G'B'$ represents gamma-compressed color components.

For HDTV (BT.709 standard):

$$Y'_{709} = 0.2125R' + 0.7154G' + 0.0721B'$$

This formula is used for modern high-definition television.

These transformations help separate brightness from color, making video compression and transmission more efficient.

➤ YCbCr and Color Conversion

In NTSC and PAL video systems, the chroma signals (color information) were filtered and superimposed on the **Y' luma signal**. This allowed older black-and-white TVs to ignore the color information, while color TVs could process it correctly.

In modern digital video and still image compression, the **YCbCr color space** is commonly used. It is closely related to YUV but uses different scale factors to fit within the 8-bit digital signal range.

- **Y' (Luma)** represents brightness and is adjusted within the range **[16...235]** in video formats.
- **Cb and Cr (Chroma)** represent color differences and are scaled within **[16...240]** in video formats.
- For **JPEG images**, the full 8-bit range **[0...255]** is used for all values.

The conversion from RGB to YCbCr is given by the matrix:

$$\begin{bmatrix} Y' \\ Cb \\ Cr \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ -0.168736 & -0.331264 & 0.5 \\ 0.5 & -0.418688 & -0.081312 \end{bmatrix} \begin{bmatrix} R' \\ G' \\ B' \end{bmatrix} + \begin{bmatrix} 0 \\ 128 \\ 128 \end{bmatrix}$$

Here, **R'G'B'** are the **gamma-compressed** 8-bit RGB values.

- This conversion is mainly handled by software when opening or saving an image.
- It is useful for compression and color processing but may require careful handling in **image deblocking** (removing artifacts from compressed images).

HSV Color Space

Another commonly used color space is **HSV (Hue, Saturation, Value)**.

- **Hue** represents the color type.
- **Saturation** measures color intensity.
- **Value** represents brightness (similar to luma).

This model is useful for intuitive color adjustments and computer vision applications.

➤ HSV and Color Ratios

HSV (Hue, Saturation, Value) Color Space

- **HSV** is another way to represent colors by transforming RGB values into:
 - **Hue (H)**: The color type, represented as an angle around a color wheel.
 - **Saturation (S)**: The intensity of the color, with higher values indicating purer colors and lower values appearing more gray.
 - **Value (V)**: The brightness of the color, which can be defined as either the mean or the maximum RGB value.
- **Usage**:
 - HSV is useful in **graphics applications** like color picking and editing.
 - It approximates the **Munsell color chart**, a system for organizing colors based on human perception.
 - In HSV images, **saturation** is often shown in grayscale (darker = more saturated), while **hue** is represented in full color.

Using Color Ratios for Image Processing

- If an algorithm should only affect **luminance (brightness)** and not **saturation or hue**, an alternative approach is to use:
 - **Yxy color space**, which separates luminance and chromaticity.
 - **Simpler color ratios**, defined as:

$$r = \frac{R}{R + G + B}, \quad g = \frac{G}{R + G + B}, \quad b = \frac{B}{R + G + B}$$

- These ratios help maintain color balance while modifying brightness.
- **Application:**
 - After adjusting the luma component (e.g., through **histogram equalization**), colors can be restored by multiplying each color ratio by the ratio of new-to-old luma values.
 - There are many color models, but in practice, their differences may not matter much depending on the application.
 - The *Lab color space**, designed to align with human perception, is also similar in purpose to HSV.
 - For more details, Charles Poynton's **Color FAQ** (available online) is a helpful resource.

2.3.3 Image Compression

1. Final Step in Camera Processing

- Before storing an image, most cameras **compress** the data (except when using formats like RAW or PNG, which are lossless).

2. YCbCr Conversion

- Images and videos are converted from **RGB to YCbCr** to **prioritize luminance (Y)** over color details (Cb, Cr), since human eyes detect brightness better than color.
- **In video:** Color components (Cb, Cr) are **reduced horizontally** (subsampling).
- **In images (JPEG):** Color components are **reduced both horizontally and vertically** to save space.

3. Transforming the Image

- After reducing color data, the image is **split into blocks** for further compression.
- The **Discrete Cosine Transform (DCT)** is used, which is similar to the **Discrete Fourier Transform (DFT)** but works with real values.
- **DCT helps** by concentrating most of the image's energy into fewer values, making compression more efficient.
- This method is used in **JPEG (images) and MPEG (videos)**.

Image compression reduces file size by focusing on important details (luminance) while reducing less noticeable details (color), using **YCbCr conversion, subsampling, and DCT transformation**.

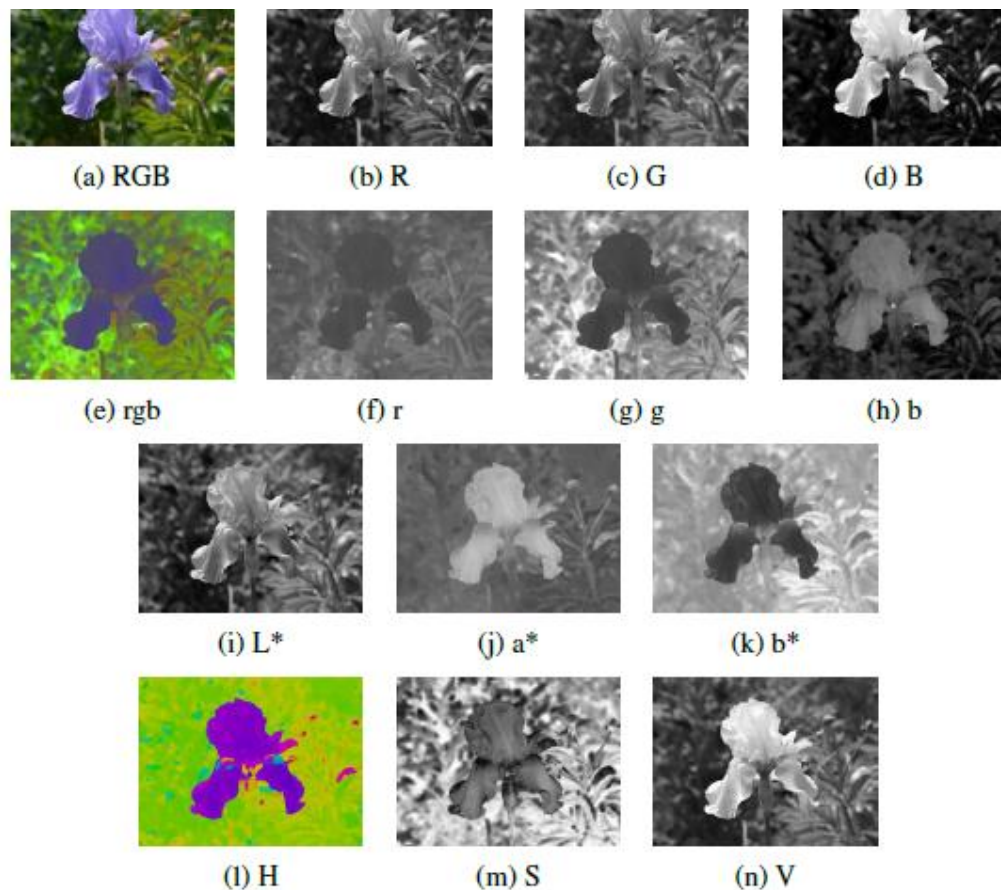


Figure 2.33 *Color space transformations: (a–d) RGB; (e–h) rgb. (i–k) $L^*a^*b^*$; (l–n) HSV. Note that the rgb, $L^*a^*b^*$, and HSV values are all re-scaled to fit the dynamic range of the printed page.*



Figure 2.34 *Image compressed with JPEG at three quality settings. Note how the amount of block artifact and high-frequency aliasing (“mosquito noise”) increases from left to right.*

Image and Video Compression

1. Compression Variants

- Modern formats like **AV1** use smaller block sizes (e.g., **4×4 or 2×2 blocks**) for better compression.
- Alternative techniques include **wavelets (JPEG 2000)** and **lapped transforms** (used in JPEG XR).

2. Quantization and Coding

- After applying a transform (like **DCT**), the values are converted into small integers.
- These are then **compressed using Huffman coding or arithmetic coding**.
- **DC coefficients (low-frequency components)** are predicted from previous blocks.
- **Quality settings** in JPEG control the **step size in quantization**.

3. Motion Compensation in Videos

- **Motion compensation** is used in video compression to **predict** pixel values from previous frames.
- **MPEG uses 16×16 motion blocks**, while newer standards use **adaptive block sizes and sub-pixel motion compensation** for better efficiency.
- **I-frames** (independent frames) are mixed with **P-frames** (predicted frames) for **better error recovery and random access**.

4. Measuring Compression Quality

- **Peak Signal-to-Noise Ratio (PSNR)** is commonly used to evaluate compression quality.
- PSNR is derived from the **Mean Square Error (MSE)**, which measures the difference between the original and compressed image.

Compression improves storage and transmission by reducing file size while preserving important details, using techniques like **block transforms, quantization, coding, and motion compensation**.

The quality of a compression algorithm is usually reported using its *peak signal-to-noise ratio* (PSNR), which is derived from the average *mean square error*,

$$MSE = \frac{1}{n} \sum_{\mathbf{x}} \left[I(\mathbf{x}) - \hat{I}(\mathbf{x}) \right]^2, \quad (2.118)$$

where $I(\mathbf{x})$ is the original uncompressed image and $\hat{I}(\mathbf{x})$ is its compressed counterpart, or equivalently, the *root mean square error* (RMS error), which is defined as

$$RMS = \sqrt{MSE}. \quad (2.119)$$

The PSNR is defined as

$$PSNR = 10 \log_{10} \frac{I_{\max}^2}{MSE} = 20 \log_{10} \frac{I_{\max}}{RMS}, \quad (2.120)$$

where $I_{\max}$ is the maximum signal extent, e.g., 255 for eight-bit images.

While this is just a high-level sketch of how image compression works, it is useful to understand so that the artifacts introduced by such techniques can be compensated for in various computer vision applications.

3.1 Point operators

The simplest kinds of image processing transforms are point operators, where each output pixel's value depends on only the corresponding input pixel value (plus, potentially, some

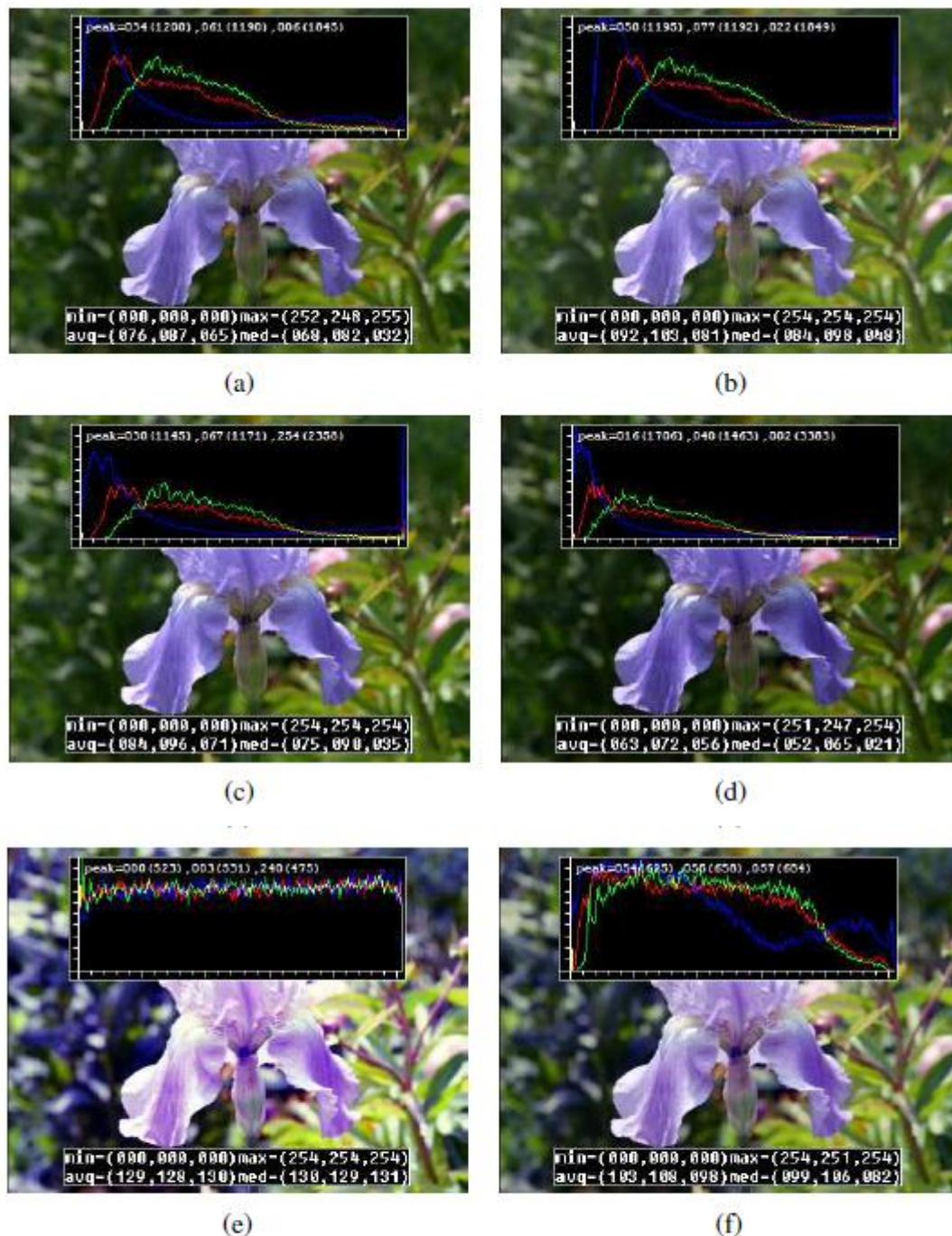


Figure 3.2 Some local image processing operations: (a) original image along with its three color (per-channel) histograms; (b) brightness increased (additive offset, $b = 16$); (c) contrast increased (multiplicative gain, $a = 1.1$); (d) gamma (partially) linearized ($\gamma = 1.2$); (e) full histogram equalization; (f) partial histogram equalization.

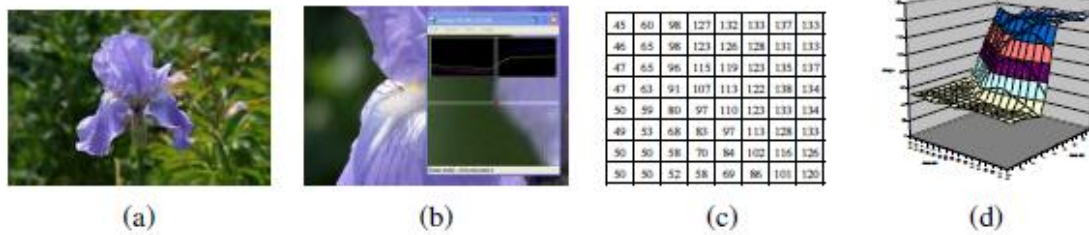


Figure 3.3 Visualizing image data: (a) original image; (b) cropped portion and scanline plot using an image inspection tool; (c) grid of numbers; (d) surface plot. For figures (c)–(d), the image was first converted to grayscale.

Image Processing Operators

1. Basic Adjustments

- Image processing includes **brightness and contrast adjustments, color correction, and transformations**.
- These are also called **point processes**, as they modify individual pixel values independently.

2. Types of Image Processing Operations

- Simple point operators** like **brightness scaling and image addition** adjust image intensity.
- Color manipulation** changes the appearance of colors in an image.
- Image compositing and matting** help in combining multiple images, important in **photography and graphics**.
- Histogram equalization** enhances image contrast by spreading out pixel intensity levels.

3. Application in Image Enhancement

- These techniques help in improving **exposure and contrast**, making images clearer and visually appealing.

Image processing techniques allow modification of brightness, contrast, colors, and composition, making them essential for **photography, graphics, and computer vision**.

3.1.1 Pixel transforms:

Definition of Pixel Transforms

- A pixel transform is a function that takes an input image and produces an output image.
- It is represented mathematically as:

$$g(x) = h(f(x))$$

or

$$g(x) = h(f_0(x), \dots, f_n(x))$$

- Here, x represents the pixel locations in an image.

Discrete vs. Continuous Domain

- In **continuous** form, functions operate over a range (scalar or vector-valued).
- In **discrete** form (sampled images), operations apply to specific pixel locations:

$$g(i, j) = h(f(i, j))$$

Representation of Images

- Images can be represented by **color (appearance)**, as a **grid of numbers**, or as a **2D surface plot**.

Common Pixel Transform Operations

- Two frequently used **point processes**:

$$g(x) = af(x) + b$$

- **a (gain)**: Controls **contrast**.
- **b (bias)**: Controls **brightness**.

Pixel transforms modify images by applying functions to pixel values, commonly adjusting brightness and contrast for image enhancement.

Gain and Bias Parameters

- **Gain (a)** controls **contrast**.
- **Bias (b)** controls **brightness**.

- These parameters can vary spatially:

$$g(x) = a(x)f(x) + b(x)$$

- Used in **graded density filters** to darken selective areas, like the sky.

Linear Operations

- **Multiplicative gain** is a linear operation because it follows the **superposition principle**:

$$h(f_0 + f_1) = h(f_0) + h(f_1)$$

- Some operations, like **image squaring** (used for energy estimation), are **not linear**.

Linear Blending (Dyadic Operator)

- A **two-input operator** used for blending images:

$$g(x) = (1 - \alpha)f_0(x) + \alpha f_1(x)$$

- Varying α from 0 to 1 creates a **cross-dissolve effect** (used in videos, slideshows, and morphing).

Gamma Correction (Non-Linear Transform)

- Used to remove the **non-linear mapping** between input radiance and quantized pixel values.
- Gamma correction formula:

$$g(x) = [f(x)]^{1/\gamma}$$

- $\gamma \approx 2.2$ is commonly used for most digital cameras.

-
- ☐ Linear **operations** preserve the sum of inputs, like blending and brightness/contrast changes.
 - ☐ Non-linear **operations**, like **gamma correction**, adjust image tones for better visual representation.

3.1.2 Color Transforms:

While color images can be treated as arbitrary vector-valued functions or collections of independent bands, it usually makes sense to think about them as highly correlated signals with strong connections to the image formation process, sensor design and human perception.

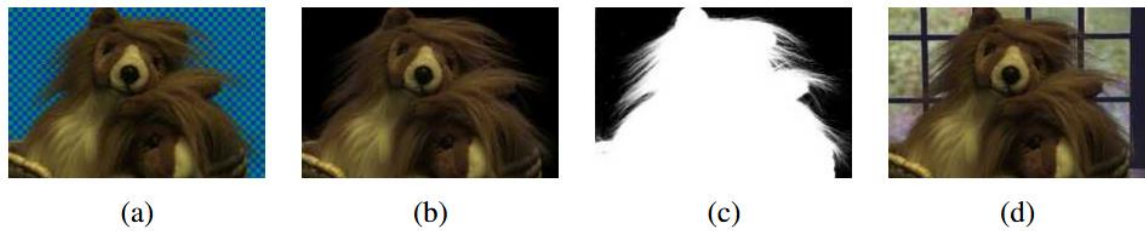


Figure 3.4 Image matting and compositing (Chuang, Curless et al. 2001) © 2001 IEEE: (a) source image; (b) extracted foreground object F ; (c) alpha matte α shown in grayscale; (d) new composite C .

1. Color Image Basics

- Color images have **three channels (RGB)** that are closely related.
- Adjusting colors affects **image brightness, contrast, and overall appearance**.

2. Effects of Brightening

- Simply adding a constant value to all RGB channels **increases intensity** but can also **change hue and saturation**.
- To **brighten** without affecting color balance, use **chromaticity coordinates or color ratios**.

3. Color Balancing Techniques

- **Fixes color distortions** caused by different lighting conditions (e.g., incandescent light).
- Methods include:
 - Multiplying each **RGB channel** by a different scale factor.
 - Converting to **XYZ color space**, adjusting the white point, and converting back to **RGB** using a **3×3 transformation matrix**.
- Be careful when **adjusting brightness** to **preserve natural colors**.
- Use **chromaticity adjustments and color balancing** for **accurate color representation**.

3.1.3 Matting and Compositing

1. Matting and Compositing

- **Matting** is the process of cutting out a foreground object from an image.
- **Compositing** is the process of placing the cut-out object onto a new background without noticeable artifacts.

2. Alpha-Matted Image

- An alpha-matted image is an intermediate version of the cut-out object before placing it on a new background.
- It contains **four channels**: Red (R), Green (G), Blue (B), and Alpha (A).

3. Alpha Channel (Opacity Control)

- The **alpha channel (A)** controls transparency at each pixel.
- **Fully opaque** pixels (inside the object) have $\alpha = 1$.
- **Fully transparent** pixels (outside the object) have $\alpha = 0$.
- **Boundary pixels** have values between 0 and 1, creating a smooth transition and avoiding jagged edges.

This technique helps blend objects naturally into new scenes without harsh edges.

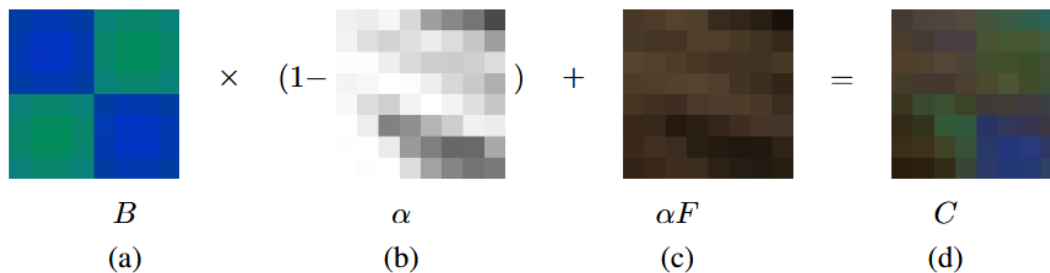


Figure 3.5 Compositing equation $C = (1 - \alpha)B + \alpha F$. The images are taken from a close-up of the region of the hair in the upper right part of the lion in Figure 3.4.

Over Operator in Image Compositing

Definition & Origin

- The **over operator** is used to composite a foreground image over a background image.
- Proposed by **Porter and Duff (1984)** and later studied by **Blinn (1994a; 1994b)**.
- The formula used is:

$$C = (1 - \alpha)B + \alpha F$$

Functionality

- The operator **attenuates** the influence of the background (BBB) by a factor of $(1 - \alpha)$.
- It then **adds** the color and opacity values of the foreground layer (FFF).

Pre-Multiplied Alpha Representation

- In many cases, storing and manipulating **pre-multiplied** alpha values (αF) is convenient.
- **Advantages of Pre-Multiplied RGBA Representation** (Blinn, 1994b):
 - Easier to blur or resample (e.g., rotate) images.
 - No extra complications when handling alpha-matted images.
 - Each RGBA band is treated independently.

Local Color Consistency & Un-Multiplied Representation

- When **matting** using local color consistency (Ruzon and Tomasi 2000; Chuang, Curless et al. 2001), using **un-multiplied** foreground colors (F) is preferred.
- This ensures:
 - Colors remain **constant or vary slowly** near object edges.



Figure 3.6 *An example of light reflecting off the transparent glass of a picture frame (Black and Anandan 1996) © 1996 Elsevier. You can clearly see the woman's portrait inside the picture frame superimposed with the reflection of a man's face off the glass.*

Additional Compositing Operations & Matting: *Additional Compositing Operations*

- Porter and Duff (1984) describe other operations useful for photo editing and visual effects.

Transparent Motion & Light Reflection

- When light reflects off transparent glass:
 - The light passing through and light reflecting off the glass are added together.
 - This model helps analyze transparent motion (Black & Anandan 1996; Szeliski, Avidan, & Anandan 2000).

- **Transparent motion** occurs when scenes are observed from a moving camera (Section 9.4.2).

Matting Process

- **Matting:** Extracting **foreground, background, and alpha matte values** from images.
- **Historical Perspective:**
 - **Smith & Blinn (1996):** Traditional **blue-screen matting** techniques.
 - **Toyama, Krumm et al. (1999):** Review on **difference matting**.

Modern Computational Photography Techniques

- Increased research in **natural image matting**:
 - **Ruzon & Tomasi (2000), Chuang, Curless et al. (2001), Wang & Cohen (2009), Xu, Price et al. (2017).**
 - Aim: Extract mattes from **single images** (Figure 3.4a) or **extended video sequences**.
 - **Chuang, Agarwala et al. (2002)** explore techniques for video sequences.

3.1.4 Histogram Equalization

What is Histogram Equalization?

- It is a technique to **automatically adjust brightness and contrast** in an image.
- The goal is to **spread out pixel intensity values** to use the full range of brightness levels.

Why is it Needed?

- Basic brightness and gain controls (from **Section 3.1.1**) can improve an image, but choosing the best values manually is hard.
- Some images have **too many dark or bright pixels** and **few mid-tone values**, making them look unbalanced.
- A better approach is to **adjust brightness and contrast dynamically** to improve visibility.

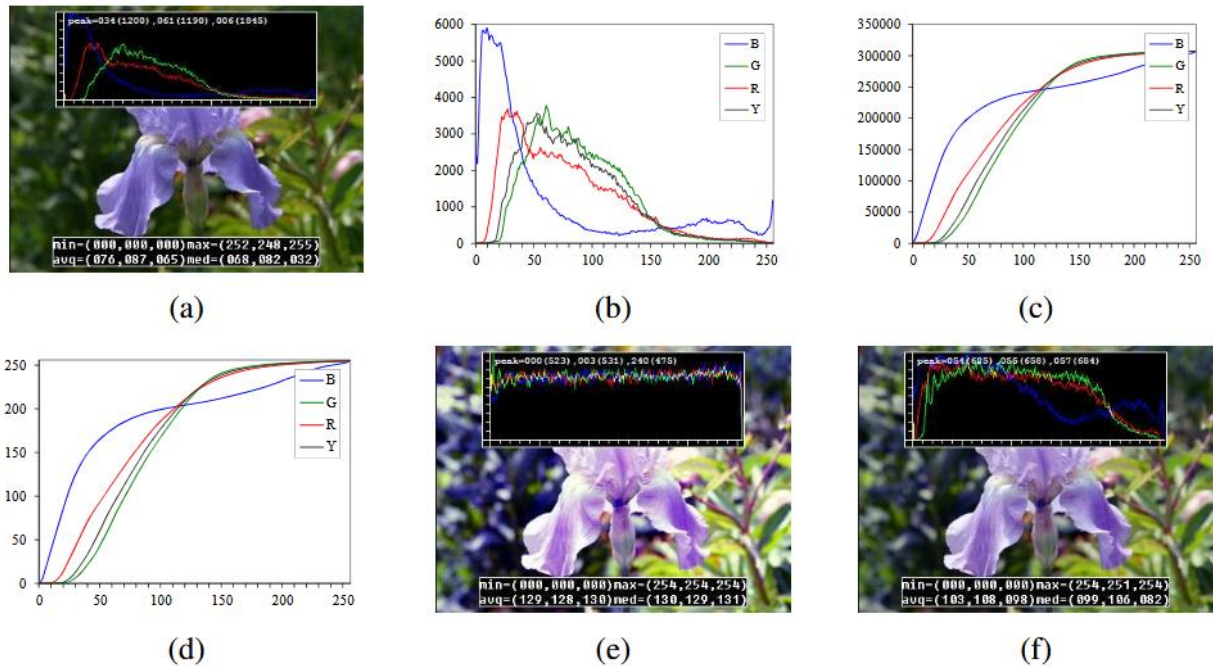


Figure 3.7 Histogram analysis and equalization: (a) original image; (b) color channel and intensity (luminance) histograms; (c) cumulative distribution functions; (d) equalization (transfer) functions; (e) full histogram equalization; (f) partial histogram equalization.

How Does it Work?

1. Visualizing Lightness Distribution:

- We can **plot a histogram** to show how pixel brightness values are distributed.
- From the histogram, we can find:
 - **Minimum and maximum brightness** values.
 - **Average brightness** level.
- Example: If an image has **too many dark and bright pixels**, but **not enough mid-tones**, adjustments are needed.

2. Adjusting the Brightness Range:

- We aim to **brighten dark areas** and **darken bright areas** while keeping a **balanced contrast**.
- This ensures the full **dynamic range** is used.

3. Histogram Equalization Technique:

- **Goal:** Create a brightness mapping function $f(I)$ that makes the histogram more uniform (flat).
- **How?** By using the **Cumulative Distribution Function (CDF)**:
 - First, calculate the **cumulative sum** of the histogram values.

- Then, use this sum to **remap intensity values** to evenly spread out the brightness.

Analogy – Understanding with Grades

- Think of an image histogram like **exam scores in a class**:
 - The histogram represents **how many students got each grade**.
 - The **CDF** helps determine a student's **percentile** (e.g., if a student is in the 75th percentile, they scored higher than 75% of classmates).
 - Similarly, we use the **CDF in histogram equalization** to **redistribute pixel brightness values** more evenly.
- Histogram equalization **automatically improves contrast** in an image.
- It works by **spreading out intensity values** using the **Cumulative Distribution Function (CDF)**.
- This technique is useful for **enhancing poorly lit images** and ensuring a **balanced brightness range**.

$$c(I) = \frac{1}{N} \sum_{i=0}^I h(i) = c(I - 1) + \frac{1}{N} h(I),$$

Mapping Pixels Using CDF

- **N = Total pixels** in the image (or students in a class, in our analogy).
- Each pixel's intensity (or a student's grade) is converted to a **percentile using the Cumulative Distribution Function (CDF)**.
- When working with **8-bit images** (grayscale images), the values range from **0 to 255**.

Applying Histogram Equalization

- The transformation **$f(I) = c(I)$** is applied to adjust pixel brightness.
- This results in a **flat histogram**, meaning pixel values are more evenly distributed.
- However, the image may look **dull or washed out** due to **reduced contrast**.

Improving the Contrast

- To fix this, a **partial adjustment** is used instead of full equalization: $f(I) = \alpha c(I) + (1 - \alpha)I$
- This **blends the original values with the equalized values**, keeping some of the **natural grayscale balance**.

- As shown in **Figure 3.7f**, this approach creates a more visually appealing result.

Potential Problems & Solutions

1. **Increased Noise in Dark Areas**
 - Histogram equalization can **amplify noise**, making dark areas look grainy.
2. **Maintaining Image Contrast**
 - Alternative techniques exist to **keep contrast and sharpness** while adjusting brightness.

➤ **Locally Adaptive Histogram Equalization**

Limitations of Global Histogram Equalization

- Global histogram equalization **adjusts the entire image uniformly**.
- Some images, like **Figure 3.8a**, have **varying brightness levels** across different regions.
- A single equalization curve **may not work well for all parts of the image**.

Dividing Image into Blocks for Local Equalization

- Instead of using a **single adjustment**, the image is divided into **$M \times M$ pixel blocks**.
- **Each block is equalized separately** to adjust contrast locally.
- However, this method **creates visible blocky artifacts** at the boundaries (Figure 3.8b).

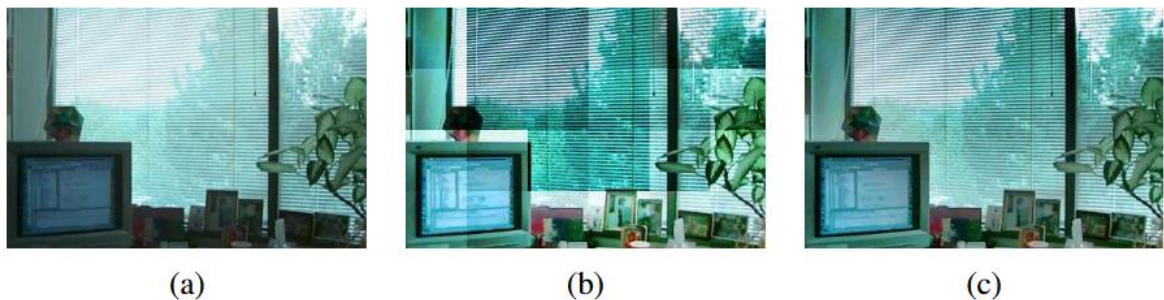


Figure 3.8 *Locally adaptive histogram equalization: (a) original image; (b) block histogram equalization; (c) full locally adaptive equalization.*

Reducing Blocky Artifacts

1. **Moving Window Approach**
 - Instead of using fixed blocks, a sliding window ($M \times M$) is moved across the image.
 - The histogram is updated for each new pixel that enters or leaves the window.
 - This is slow but can be optimized by updating only necessary histogram values.
2. **Adaptive Histogram Equalization (AHE)**
 - AHE applies block-based equalization but smoothly blends the transitions between blocks.
 - This avoids sharp intensity changes at block boundaries.
3. **Contrast-Limited Adaptive Histogram Equalization (CLAHE)**
 - A modified version of AHE that limits the contrast to prevent over-enhancement.

- Pizer, Amburn et al. (1987) introduced CLAHE to balance contrast without amplifying noise.

Bilinear Blending for Smooth Transitions

- To **blend the four lookup functions** $\{f_{00}, f_{10}, f_{01}, f_{11}\} \setminus \{f_{00}, f_{10}, f_{01}, f_{11}\}$, **bilinear interpolation** is used.
- The formula in **Equation 3.10** calculates a **smooth transition** between adjacent blocks.
- The blending function depends on the **horizontal (s) and vertical (t) position** within a block.

$$f_{s,t}(I) = (1 - s)(1 - t)f_{00}(I) + s(1 - t)f_{10}(I) + (1 - s)tf_{01}(I) + stf_{11}(I) \quad (3.10)$$

Using Bilinear Interpolation for Smoother Results

- Instead of **blending four lookup tables** for each output pixel (which is slow), a better approach is to **blend the results of mapping a pixel through the four neighboring lookup tables**.
- This approach **smooths transitions between blocks** while maintaining efficiency.

Corner-Based Lookup Table Placement

- A variation of the algorithm places the **lookup tables at the corners of each $M \times M$ block**.
- This helps in **better distribution of pixel values** across the histogram.
- Instead of assigning each input pixel to just one lookup table, it is **distributed into four adjacent lookup tables** during the **histogram accumulation phase**.

Mathematical Representation

- The equation for histogram accumulation is:

$$h_{k,l}(I(i, j)) + = w(i, j, k, l)$$

- Here, $h_{k,l}$ represents the histogram value at lookup table (k, l) .
- $w(i, j, k, l)$ is the **bilinear weighting function** between pixel (i, j) and lookup table (k, l) .

Soft Histogramming

- This **soft histogramming technique** distributes pixel values smoothly instead of assigning them rigidly to one bin.
- It is used in **many applications**, including:
 - **SIFT (Scale-Invariant Feature Transform) feature descriptors** (Section 7.1.3).
 - **Vocabulary trees** (Section 7.1.4).

3.1.5 Tonal Adjustment

What is Tonal Adjustment?

- Tonal adjustment is a **common image processing technique** used to enhance contrast and brightness in photographs.
- It makes images look **more attractive** or **easier to interpret**.

Where is it Used?

- Found in **photo editing tools** where users can adjust contrast, brightness, and color.
- Examples can be seen in **Figures 3.2 and 3.7**.

Hands-On Practice

- Exercises **3.1, 3.6, and 3.7** help in understanding these operations by implementing them.

Advanced Techniques

- More **sophisticated methods** for tonal adjustment exist, such as:
 - **High Dynamic Range (HDR) tone mapping** (Section 10.2.1).
 - **Techniques by Bae, Paris, and Durand (2006), Reinhard, Heidrich et al. (2010).**

3.2 Linear Filtering

What is Linear Filtering?

- A **neighborhood operator** (or **local operator**) that processes a pixel based on the values of surrounding pixels.
- Used for **image enhancement** by modifying pixel values in a small region.

Applications of Neighborhood Operators

- **Adjusting tones** in specific areas of an image.
- **Filtering images** for different effects:
 - **Blurring** to soften details.
 - **Sharpening** to enhance details.
 - **Edge detection** to highlight boundaries.
 - **Noise removal** to clean up the image.
- Examples can be seen in **Figures 3.10 and 3.11b–d**.

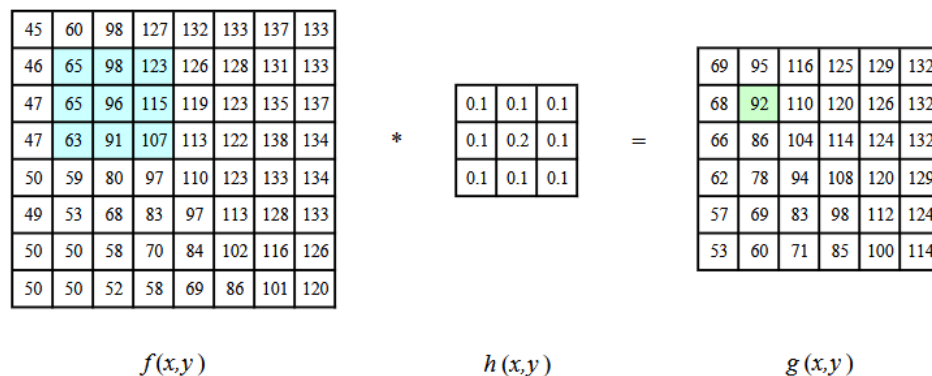


Figure 3.10 Neighborhood filtering (convolution): The image on the left is convolved with the filter in the middle to yield the image on the right. The light blue pixels indicate the source neighborhood for the light green destination pixel.

A common variant on this formula is

$$g(i, j) = \sum_{k, l} f(i - k, j - l) h(k, l) = \sum_{k, l} f(k, l) h(i - k, j - l), \quad (3.14)$$

where the sign of the offsets in f has been reversed. This is called the *convolution* operator,

$$g = f * h, \quad (3.15)$$

and h is then called the impulse response function.⁵ The reason for this name is that the kernel function, h , convolved with an impulse signal, $\delta(i, j)$ (an image that is 0 everywhere except at the origin) reproduces itself, $h * \delta = h$, whereas correlation produces the reflected signal.

3.2 Linear filtering



Figure 3.11 Some neighborhood operations: (a) original image; (b) blurred; (c) sharpened; (d) smoothed with edge-preserving filter; (e) binary image; (f) dilated; (g) distance transform; (h) connected components. For the dilation and connected components, black (ink) pixels are assumed to be active, i.e., to have a value of 1 in Equations (3.44–3.48).

$$\begin{bmatrix} 72 & 88 & 62 & 52 & 37 \end{bmatrix} * \begin{bmatrix} 1/4 & 1/2 & 1/4 \end{bmatrix} \Leftrightarrow \frac{1}{4} \begin{bmatrix} 2 & 1 & . & . & . \\ 1 & 2 & 1 & . & . \\ . & 1 & 2 & 1 & . \\ . & . & 1 & 2 & 1 \\ . & . & . & 1 & 2 \end{bmatrix} \begin{bmatrix} 72 \\ 88 \\ 62 \\ 52 \\ 37 \end{bmatrix}$$

Figure 3.12 One-dimensional signal convolution as a sparse matrix-vector multiplication, $\mathbf{g} = \mathbf{H}\mathbf{f}$.

In fact, Equation (3.14) can be interpreted as the superposition (addition) of shifted impulse response functions $h(i - k, j - l)$ multiplied by the input pixel values $f(k, l)$. Convolution has additional nice properties, e.g., it is both commutative and associative. As well, the Fourier transform of two convolved images is the product of their individual Fourier transforms.

Both correlation and convolution are *linear shift-invariant* (LSI) operators, which obey both the superposition principle (3.5),

$$h \circ (f_0 + f_1) = h \circ f_0 + h \circ f_1, \quad (3.16)$$

and the *shift invariance* principle,

$$g(i, j) = f(i + k, j + l) \Leftrightarrow (h \circ g)(i, j) = (h \circ f)(i + k, j + l), \quad (3.17)$$

which means that shifting a signal commutes with applying the operator ($\circ$ stands for the LSI operator). Another way to think of shift invariance is that the operator “behaves the same everywhere”.

Occasionally, a shift-variant version of correlation or convolution may be used, e.g.,

$$g(i, j) = \sum_{k, l} f(i - k, j - l) h(k, l; i, j), \quad (3.18)$$

where $h(k, l; i, j)$ is the convolution kernel at pixel (i, j) . For example, such a spatially varying kernel can be used to model blur in an image due to variable depth-dependent defocus.

Correlation and convolution can both be written as a matrix-vector multiplication, if we first convert the two-dimensional images $f(i, j)$ and $g(i, j)$ into raster-ordered vectors $\mathbf{f}$ and $\mathbf{g}$,

$$\mathbf{g} = \mathbf{H}\mathbf{f}, \quad (3.19)$$

where the (sparse) $\mathbf{H}$ matrix contains the convolution kernels. Figure 3.12 shows how a onedimensional convolution can be represented in matrix-vector form.

Padding (border effects)

The astute reader will notice that the correlation shown in Figure 3.10 produces a result that is smaller than the original image, which may not be desirable in many applications.⁶ This is because the neighborhoods of typical correlation and convolution operations extend beyond the image boundaries near the edges, and so the filtered images suffer from boundary effects. To deal with this, a number of different padding or extension modes have been developed for neighborhood operations (Figure 3.13):

- **zero:** set all pixels outside the source image to 0 (a good choice for alpha-matted cutout images);
- **constant** (border color): set all pixels outside the source image to a specified border value;
- **clamp** (replicate or clamp to edge): repeat edge pixels indefinitely;
- **(cyclic) wrap (repeat or tile):** loop “around” the image in a “toroidal” configuration;
- **mirror:** reflect pixels across the image edge.

extend: extend the signal by subtracting the mirrored version of the signal from the edge pixel value. In the computer graphics literature, these mechanisms are known as the wrapping mode (OpenGL) or texture addressing mode.

Figure 3.13 shows the effects of padding an image with each of the above mechanisms and then blurring the resulting padded image. As you can see, zero padding darkens the edges, clamp

(replication) padding propagates border values inward, mirror (reflection) padding preserves colors near the borders. Extension padding (not shown) keeps the border pixels fixed (during blur).

3.2.1 Separable Filtering

Separable filtering reduces computational cost in image processing by breaking a **2D convolution** into two **1D convolutions** (horizontal and vertical).

- **Standard convolution** requires K^2 operations per pixel.
- **Separable convolution** only requires $2K$ operations per pixel, making it much more efficient.
- A **kernel** is separable if it can be expressed as the **outer product** of two 1D vectors:

$$K = vh^T$$

- To check if a kernel is separable, we can:
 1. **Inspect the kernel manually.**
 2. **Use Singular Value Decomposition (SVD)** to determine if it can be factored into two 1D kernels.

Separable filtering is widely used in **Gaussian blurring**, **box filtering**, and other image processing applications due to its efficiency.

To check if a **convolution kernel** is **separable**, we can:

- **Manually inspect the kernel** or analyze its structure.
- **Use Singular Value Decomposition (SVD)**, where the 2D kernel **K** is decomposed as:

$$K = \sum \sigma_i u_i v_i^T$$

- If only **one singular value** (σ_0) is **non-zero**, the kernel is separable.
- The corresponding vectors $\sqrt{\sigma_0}u_0$ (**vertical kernel**) and $\sqrt{\sigma_0}v_0^T$ (**horizontal kernel**) define the separable components.

Separable kernels reduce computational complexity and are widely used in **image processing** for efficient filtering.

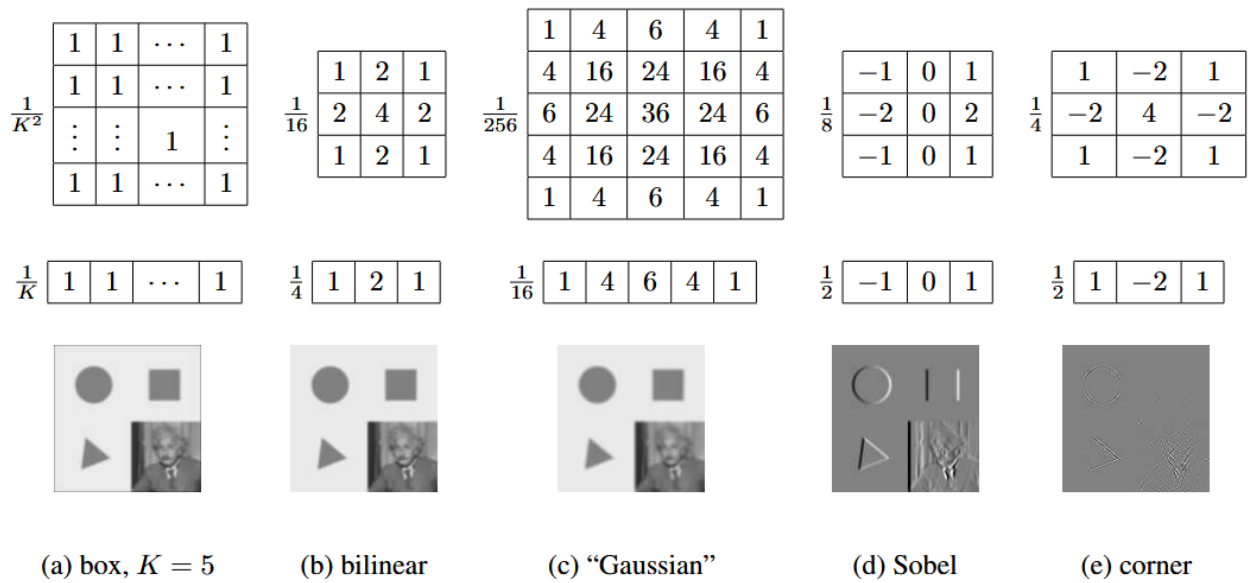


Figure 3.14 *Separable linear filters: For each image (a)–(e), we show the 2D filter kernel (top), the corresponding horizontal 1D kernel (middle), and the filtered image (bottom). The filtered Sobel and corner images are signed, scaled up by $2\times$ and $4\times$, respectively, and added to a gray offset before display.*

3.2.2 Examples of linear filtering

Linear filtering is a fundamental technique in image processing that involves applying mathematical operations to enhance or modify images. Below are some key examples:

1. Smoothing Filters (Blurring or Low-Pass Filters)

These filters reduce noise and blur an image by averaging pixel values in a local region.

- **Moving Average (Box Filter)**
 - Averages pixel values in a $K \times K$ window.
 - Implemented by convolving the image with a matrix of ones and then scaling.
 - Efficient for large kernels by using **summed area tables** for quick computation.
- **Tent Filter (Bilinear Filter)**
 - Uses a **piecewise linear function** to smooth images.
 - A 3×3 **bilinear kernel** is formed by the outer product of two linear splines.
 - Helps preserve details while reducing noise.
- **Gaussian Filter**
 - Obtained by convolving the bilinear filter with itself.
 - Provides a smoother result compared to the box filter.
 - Can be approximated by iterating box filter operations multiple times.
 - Used when **rotationally symmetric smoothing** is needed.
- **Sinc Filter (sin(x)/x Filter)**
 - Performs **ideal low-pass filtering** by preserving low frequencies while removing high frequencies.
 - Theoretical but not commonly used in real-world applications due to computational complexity.

2. Image Sharpening (Unsharp Masking)

Although smoothing filters usually blur images, they can also be used to **sharpen** them.

- **Unsharp Masking Process**
 1. Blur the image using a smoothing filter (e.g., Gaussian).
 2. Subtract the blurred image from the original to extract high-frequency details.
 3. Add this difference back to the original image to enhance sharpness.
 - Formula:

$$g_{\text{sharp}} = f + \gamma(f - h_{\text{blur}} * f)$$

- where γ controls the sharpening intensity.

□ Historical **Method (Before Digital Processing)**

- In darkroom photography, photographers created a **blurred positive negative** by misfocusing the image.
- Overlaying this with the original negative produced a sharper final image.
- The mathematical representation is:

$$g_{\text{unsharp}} = f(1 - \gamma h_{\text{blur}} * f)$$

- Though not a linear filter, this method remains effective in image enhancement.

3. Edge Detection Filters

Edge detection is crucial for **feature extraction**, **object recognition**, and **computer vision applications**.

- **Sobel Operator**
 - A **3 × 3 filter** used to detect edges in an image.
 - Combines:
 - **Horizontal central difference filter** (detects horizontal edges).
 - **Vertical tent filter** (smooths noise in the vertical direction).
 - Highlights **vertical edges** effectively.
- **Corner Detection**
 - Detects corners by analyzing **horizontal and vertical second derivatives**.
 - A simple **3 × 3 kernel** identifies corners but may also respond to diagonal edges.
 - More advanced detectors provide **rotation invariance**, improving accuracy in feature detection.

NOTE:

- ✓ **Smoothing filters** (box, bilinear, Gaussian) reduce noise and blur images.
- ✓ **Unsharp masking** sharpens images by enhancing high-frequency details.
- ✓ **Edge detection filters** (e.g., Sobel operator) identify boundaries in images.
- ✓ **Corner detectors** find key points in an image, useful for computer vision tasks.

3.2.3 Band-pass and Steerable Filters

Filters in image processing help extract specific features from images, such as edges or textures. **Band-pass filters** allow a specific range of frequencies, filtering out both very low and very high frequencies. **Steerable filters** can be adjusted to respond to features at different orientations.

Band-pass Filters:

Band-pass filters help detect edges and textures by removing unwanted frequency components.

One common approach is:

1. **Smooth the image** using a Gaussian filter:

$$G(x, y; \sigma) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

- Here, σ controls the level of blurring.

2. **Compute the second derivative** using the Laplacian operator:

$$\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$$

- This operator highlights areas with rapid intensity changes (edges, corners).

3. **Laplacian of Gaussian (LoG) filter** is directly used:

$$\nabla^2 G(x, y; \sigma) = \left(\frac{x^2 + y^2}{\sigma^4} - \frac{2}{\sigma^2} \right) G(x, y; \sigma)$$

- This filter is useful for detecting blobs and texture details.



Steerable Filters

Steerable filters can be adapted to respond to specific orientations.

1. **Directional derivative** of an image along a unit vector $\mathbf{u} = (\cos \theta, \sin \theta)$:

$$\mathbf{u} \cdot \nabla f(x, y) = \nabla_{\theta} f = (\nabla G) * f$$

- The gradient direction is controlled by θ .

2. **Smooth directional derivative filter**:

$$G_{\theta} = uG_x + vG_y = \cos \theta \frac{\partial G}{\partial x} + \sin \theta \frac{\partial G}{\partial y}$$

3. **Second-order steerable filter** (Freeman & Adelson 1991) approximates any direction efficiently:

$$G_{\theta\theta} = v^2 G_{xx} + 2uv G_{xy} + u^2 G_{yy}$$

- This allows edge detection at different orientations with fewer computations.

Applications

- **Edge and corner detection:** Enhances detection of edges in specific directions.
- **Texture analysis:** Helps analyze patterns in images.
- **Image enhancement:** Used in sharpening and feature extraction.

NOTE:

- ☐ Band-pass filters enhance features by filtering out both low and high frequencies.
- ☐ Steerable filters can be rotated to detect edges at any orientation efficiently.
- ☐ The Laplacian of Gaussian (LoG) is a common band-pass filter.
- ☐ Freeman and Adelson's steerable filters allow efficient multi-directional feature detection.