AKSHAYA INSTITUTE OF TECHNOLOGY

Lingapura, Tumkur-Koratagere Road, Tumkur-572106.

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

Module 1 Notes for

"Computer Vision"

[BCS613B]

Prepared by: -

Mr. PRADEEP S

Assistant Professor, Department of CSE.

Akshaya Institute of Technology,

Tumakuru

AKSHAYA INSTITUTE OF TECHNOLOGY

Lingapura, Obalapura Post, Koratagere Road, Tumakuru - 572106

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

VISION

To empower the students to be technically competent, innovative and self-motivated with human values and contribute significantly towards betterment of society and to respond swiftly to the challenges of the changing world.

MISSION

M1: To achieve academic excellence by imparting in-depth and competitive knowledge to the students through effective teaching pedagogies and hands on experience on cutting edge technologies.

M2: To collaborate with industry and academia for achieving quality technical education and knowledge transfer through active participation of all the stake holders.

M3: To prepare students to be life-long learners and to upgrade their skills through Centre of Excellence in the thrust areas of Computer Science and Engineering.

Program Specific Outcomes (PSOs)

After Successful Completion of Computer Science and Engineering Program Students will be able to

* Apply fundamental knowledge for professional software development as well as to acquire new skills.
* Implement disciplinary knowledge in problem solving, analyzing and decision-making abilities through different domains like database management, networking, algorithms, and programming as well as research and development.
* Make use of modern computer tools for creating innovative career paths, to become an entrepreneur or desire for higher studies.

Program Educational Objectives (PEOs)

PEO1: Graduates expose strong skills and abilities to work in industries and research organizations.

PEO3: Graduates engage in team work to function as responsible professional with good ethical behavior and leadership skills.

PEO3: Graduates engage in life-long learning and innovations in multi disciplinary areas.

Module 1: Introduction

Computer Vision is a branch of Artificial Intelligence (AI) that enables computers to interpret and extract information from images and videos, similar to human perception. It involves developing algorithms to process visual data and derive meaningful insights.

Think of how vivid the three-dimensional percept is when you look at a vase of flowers sitting on the table next to you. You can tell the shape and translucency of each petal through the subtle patterns of light and shading that play across its surface and effortlessly segment each flower from the background of the scene

The human visual system has no problem interpreting the subtle variations in translucency and shading in this photograph and correctly segmenting the object from its background.
Researchers in computer vision have been developing, in parallel, mathematical techniques for recovering the three-dimensional shape and appearance of objects in imagery.
Here, the progress in the last two decades has been rapid. We now have reliable techniques for accurately computing a 3D model of an environment from thousands of partially overlapping photographs
Why is vision so difficult? In part, it is because it is an inverse problem, in which we seek to recover some unknowns given insufficient information to fully specify the solution computer vision is being used today in a wide variety of *real-world applications*, which include:

- **Optical character recognition (OCR)**: reading handwritten postal codes on letters (Figure 1.4a) and automatic number plate recognition (ANPR);
- **Machine inspection**: rapid parts inspection for quality assurance using stereo vision with specialized illumination to measure tolerances on aircraft wings or auto body parts (Figure 1.4b) or looking for defects in steel castings using X-ray vision;
- **Retail**: object recognition for automated checkout lanes and fully automated stores
- **Warehouse logistics**: autonomous package delivery and pallet-carrying “drives” and parts picking by robotic manipulators (Figure 1.4c;)
- **Medical imaging**: registering pre-operative and intra-operative imagery (Figure 1.4d) or performing long-term studies of people’s brain morphology as they age;
- **Self-driving vehicles**: capable of driving point-to-point between cities (Figure 1.4) as well as autonomous flight
- **3D model building (photogrammetry):** fully automated construction of 3D models from aerial and drone photographs (Figure 1.4f);
- **Match move**: merging computer-generated imagery (CGI) with live action footage by tracking feature points in the source video to estimate the 3D camera motion and shape of

the environment. Such techniques are widely used in Hollywood, e.g., in movies such as Jurassic Park; they also require the use of

- **Motion capture (mocap):** using retro-reflective markers viewed from multiple cameras or other vision-based techniques to capture actors for computer animation;
- **Surveillance**: monitoring for intruders, analysing highway traffic and monitoring pools for drowning victims
- **Fingerprint recognition and biometrics:** for automatic access authentication as well as forensic applications.

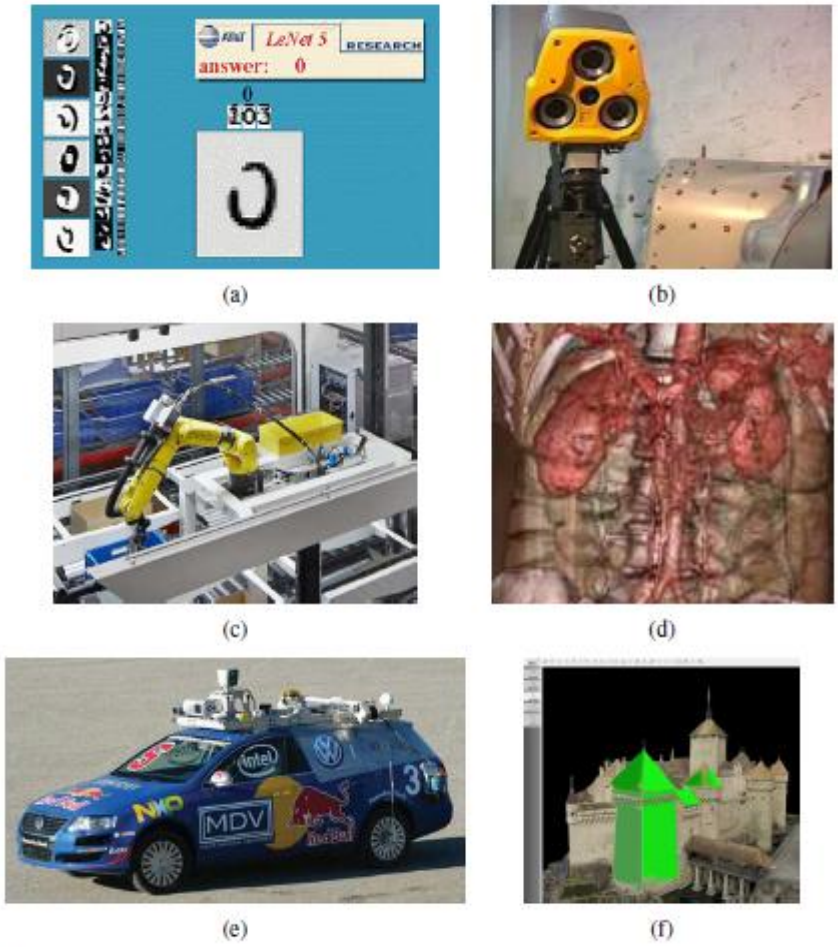


Figure 1.4 *Some industrial applications of computer vision: (a) optical character recognition (OCR), http://yann.lecun.com/exdb/lenet; (b) mechanical inspection, http://www.cognitens.com; (c) warehouse picking, https://covariant.ai; (d) medical imaging, http://www.clarontech.com; (e) self-driving cars, (Montemerlo, Becker* et al.

Consumer-level applications of Computer Vision:

- **Stitching**: turning overlapping photos into a single seamlessly stitched panorama (Figure1.5a), • Exposure bracketing: merging multiple exposures taken under challenging lighting conditions (strong sunlight and shadows) into a single perfectly exposed image (Figure1.5b),
- **Morphing**: turning a picture of one of your friends into another, using a seamless morph transition (Figure 1.5c);
- **3D modelling**: converting one or more snapshots into a 3D model of the object or person you are photographing (Figure 1.5d),
- **Video match move and stabilization**: inserting 2D pictures or 3D models into your videos by automatically tracking nearby reference points or using motion estimates to remove shake from your videos

- **Photo-based walkthroughs**: navigating a large collection of photographs, such as the interior of your house, by flying between different photos in 3D
- **Face detection**: for improved camera focusing as well as more relevant image searching Visual authentication: automatically logging family members onto your home computer as they sit down in front of the webcam

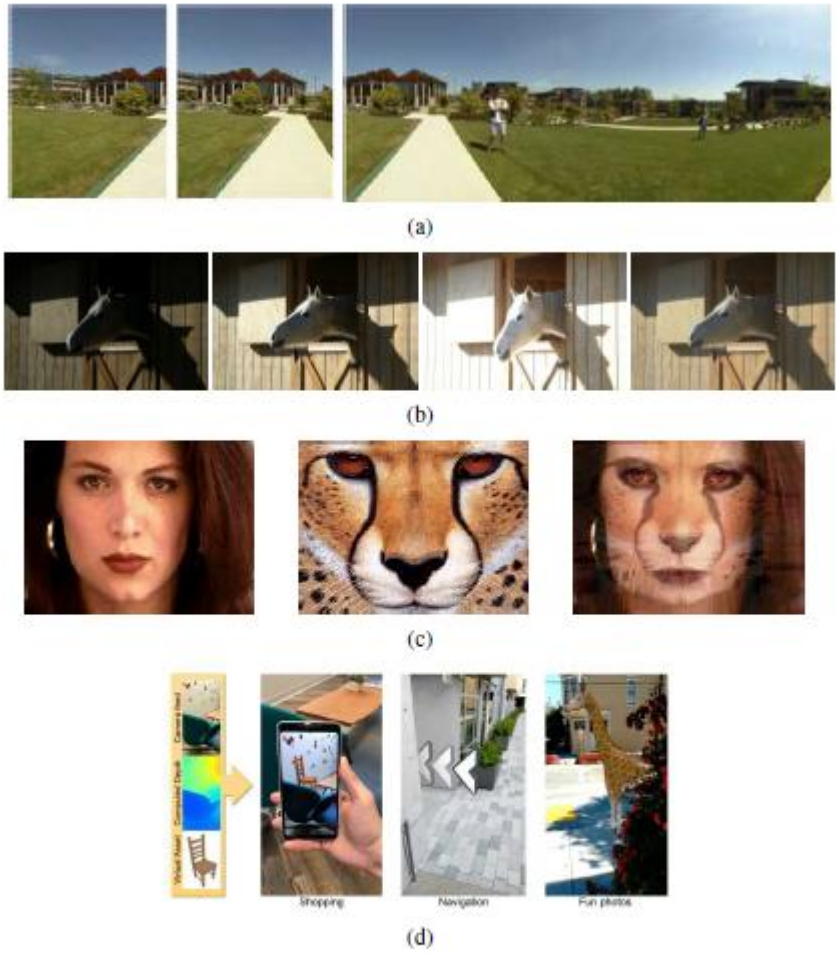
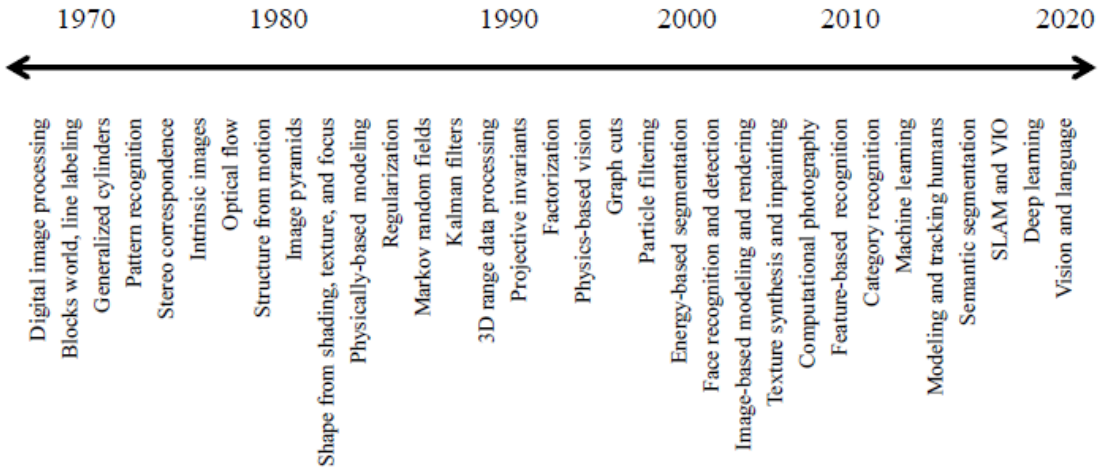


Figure 1.5 *Some consumer applications of computer vision: (a) image stitching: merging different views (Szeliski and Shum 1997) © 1997 ACM; (b) exposure bracketing: merging different exposures; (c) morphing: blending between two photographs (Gomes, Darsa* et al. *1999) © 1999 Morgan Kaufmann; (d) smartphone augmented reality showing real-time depth occlusion effects (Valentin, Kowdle* et al. *2018) © 2018 ACM.*

A brief history:



- **1970s**. When computer vision first started out in the early 1970s, it was viewed as the visual perception component of an ambitious agenda to mimic human intelligence and to endow robots with intelligent behaviour. Three-dimensional modelling of non-polyhedral objects was also being studied One popular approach used generalized cylinders, i.e., solids of revolution and swept closed curves

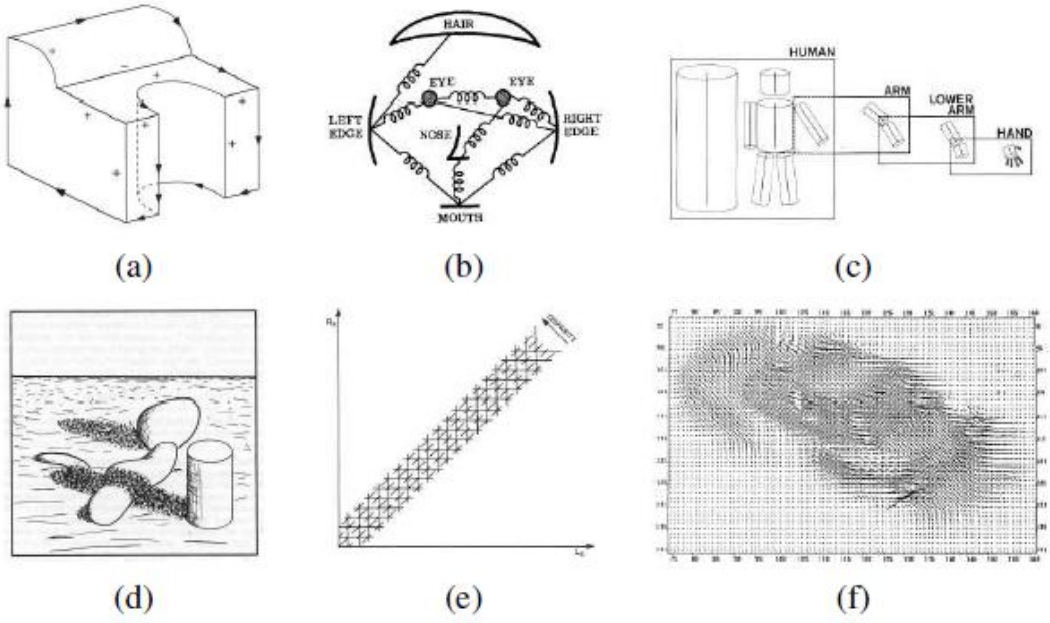


Figure 1.7 *Some early (1970s) examples of computer vision algorithms: (a) line labeling (Nalwa 1993) © 1993 Addison-Wesley, (b) pictorial structures (Fischler and Elschlager 1973) © 1973 IEEE, (c) articulated body model (Marr 1982) © 1982 David Marr, (d) intrinsic images (Barrow and Tenenbaum 1981) © 1973 IEEE, (e) stereo correspondence (Marr 1982) © 1982 David Marr, (f) optical flow (Nagel and Enkelmann 1986) © 1986 IEEE.*

- **1980s**. Image pyramids started being widely used to perform tasks such as image blending (Figure 1.8a) and coarse-to-fine correspondence search The use of stereo as a quantitative shape cue was extended by a wide variety of shape from- X techniques, including shape from shading

- **1990s**. A lot of the initial activity was directed at projective reconstructions, which did not require knowledge of camera calibration Simultaneously, factorization techniques were developed to solve efficiently problems for which orthographic camera approximations were applicable Multi-view stereo algorithms (Figure 1.9c) that produce complete 3D surfaces were also an active topic of research. Tracking algorithms also improved a lot, including contour tracking using active contours such as snakes particle filters and level sets as well as intensity-based (direct) techniques
  Texture synthesis (Figure 1.10d) quilting and in painting) are additional topics that can be classified as computational photography techniques, since they re-combine input image samples to produce new photographs.

- **2010s**. This trend was enabled by the development of high-quality large-scale annotated datasets such as ImageNet. Another major trend was the dramatic increase in computational power available from the development of general purpose (data-parallel) algorithms on graphical processing units (GPGPU).

Photometric image formation:

To produce an image, the scene must be illuminated with one or more light sources. Light sources can generally be divided into point and area light sources.
A point light source originates at a single location in space (e.g., a small light bulb), potentially at infinity (e.g., the Sun). the Sun may have to be treated as an area light source.) In addition to its location, a point light source has an intensity and a colour spectrum, i.e., a distribution over wavelengths L(λ). The intensity of a light source falls off with the square of the distance between the source and the object being lit, because the same light is being spread over a larger (spherical) area. A light source may also have a directional falloff (dependence), but we ignore this in our simplified model. Area light sources are more complicated. A simple area light source such as a fluorescent ceiling light fixture with a diffuser can be modelled as a finite rectangular area emitting light equally in all directions
This representation maps incident light directions $\hat{\mathbf{v}}$ to color values (or wavelengths), and is equivalent to assuming that all light sources are at infinity

$$L(\hat{\mathbf{v}}; \lambda),$$

Reflectance and shading:

When light hits an object's surface, it is scattered and reflected Many different models have been developed to describe this interaction. In this section, we first describe the most general form, the bidirectional reflectance distribution function, and then look at some more specialized models, including the diffuse, specular, and Phong shading models.

The Bidirectional Reflectance Distribution Function (BRDF):

The most general model of light scattering is the bidirectional reflectance distribution function (BRDF).8 Relative to some local coordinate frame on the surface, the BRDF is a four-dimensional function that describes how much of each wavelength arriving at an incident direction $\hat{v}i$ is emitted in a reflected direction $\hat{v}r$ (Figure 2.15b). The function can be written in terms of the angles of the incident and reflected directions relative to the surface frame as

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

The BRDF is reciprocal, i.e., because of the physics of light transport, you can interchange the roles of $\hat{v}i$ and $\hat{v}r$ and still get the same answer.

Most surfaces are isotropic, i.e., there are no preferred directions on the surface as far as light transport is concerned.

For an isotropic material, we can simplify the BRDF to

$$f_r(\theta_i, \theta_r, |\phi_r - \phi_i|; \lambda) \quad \text{or} \quad f_r(\hat{\mathbf{v}}_i, \hat{\mathbf{v}}_r, \hat{\mathbf{n}}; \lambda),$$

as the quantities Θi, Θr, and Φr - Φi can be computed from the directions $\hat{v}i$, $\hat{v}r$ and $\hat{n}$.

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

$$L_r(\hat{\mathbf{v}}_r; \lambda) = \int L_i(\hat{\mathbf{v}}_i; \lambda) f_r(\hat{\mathbf{v}}_i, \hat{\mathbf{v}}_r, \hat{\mathbf{n}}; \lambda) \cos^+ \theta_i \, d\hat{\mathbf{v}}_i, \tag{2.84}$$

where

$$\cos^+ \theta_i = \max(0, \cos \theta_i). \tag{2.85}$$

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

$$L_r(\hat{\mathbf{v}}_r; \lambda) = \sum_i L_i(\lambda) f_r(\hat{\mathbf{v}}_i, \hat{\mathbf{v}}_r, \hat{\mathbf{n}}; \lambda) \cos^+ \theta_i. \tag{2.86}$$

Diffuse reflection:

The diffuse component scatters light uniformly in all directions and is the phenomenon we most normally associate with shading, e.g., the smooth (non-shiny) variation of intensity with surface normal that is seen when observing
a statue.
Diffuse reflection also often imparts a strong body color to the light, as it is caused by selective absorption and re emission of light inside the object's material
While light is scattered uniformly in all directions, i.e., the BRDF is constant

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

The amount of light depends on the angle between the incident light direction and the surface normal Θi.
This is because the surface area exposed to a given amount of light becomes larger at oblique angles, becoming completely self-shadowed as the outgoing surface normal points away from the light.
The shading equation for diffuse reflection can thus be written as

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

Where

$$[\hat{\mathbf{v}}_i \cdot \hat{\mathbf{n}}]^+ = \max(0, \hat{\mathbf{v}}_i \cdot \hat{\mathbf{n}}).$$

Specular reflection:

The second major component of a typical BRDF is specular (gloss or highlight) reflection, which depends strongly on the direction of the outgoing light. Incident light rays are reflected in a direction that is rotated by 180° around the surface normal $\hat{n}$

we can compute the specular reflection direction $\hat{s}i$ as

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

Phong shading:

Combined the diffuse and specular components of reflection with another term, which he called the ambient illumination. This term accounts for the fact that objects are generally illuminated not only by point light sources but also by a general diffuse illumination corresponding to inter-reflection or distant sources, such as the blue sky.

In the Phong model, the ambient term does not depend on surface orientation, but depends on the color of both the ambient illumination La(λ) and the object ka(λ),

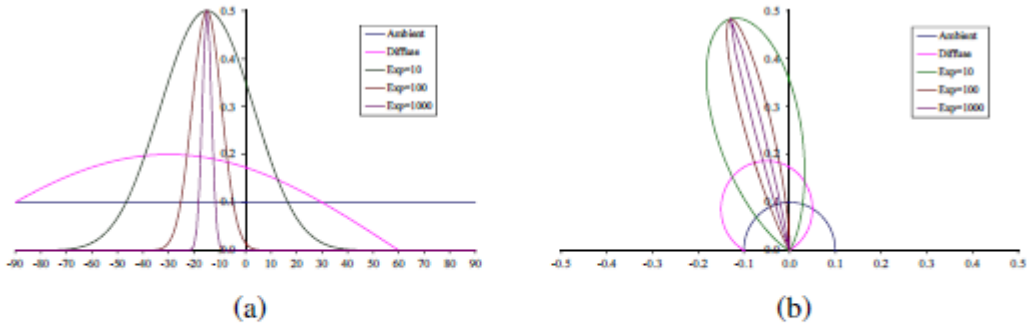


Figure 2.18 *Cross-section through a Phong shading model BRDF for a fixed incident illumination direction: (a) component values as a function of angle away from surface normal; (b) polar plot. The value of the Phong exponent k_e is indicated by the "Exp" labels and the light source is at an angle of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

typical set of Phong shading model components as a function of the angle away from the surface normal

Typically, the ambient and diffuse reflection color distributions ka(λ) and kd(λ) are the same, since they are both due to sub-surface scattering (body reflection) inside the surface material .The specular reflection distribution ks(λ) is often uniform (white),

since it is caused by interface reflections that do not change the light color

Di-chromatic reflection model:

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

$$\begin{aligned} L_r(\hat{\mathbf{v}}_r; \lambda) &= L_i(\hat{\mathbf{v}}_r, \hat{\mathbf{v}}_i, \hat{\mathbf{n}}; \lambda) + L_b(\hat{\mathbf{v}}_r, \hat{\mathbf{v}}_i, \hat{\mathbf{n}}; \lambda) \\ &= c_i(\lambda) m_i(\hat{\mathbf{v}}_r, \hat{\mathbf{v}}_i, \hat{\mathbf{n}}) + c_b(\lambda) m_b(\hat{\mathbf{v}}_r, \hat{\mathbf{v}}_i, \hat{\mathbf{n}}), \end{aligned}$$

the radiance of the light reflected at the interface, Li, and the radiance reflected at the surface body, Lb.

Global illumination:

The simple shading model presented thus far assumes that light rays leave the light sources, bounce off surfaces visible to the camera, thereby changing in intensity or color, and arrive at the camera. Two methods have traditionally been used to model such effects. If the scene is mostly specular (the classic example being scenes made of glass objects and mirrored or highly polished balls), the preferred approach is ray tracing or path tracing Combinations of the two techniques have also been developed as well as more general light transport techniques for simulating effects such as the caustics cast by rippling water. primary contribution can then be computed using the simple shading equations presented previously for all light sources that are visible for that surface element.

Optics:

Once the light from a scene reaches the camera, it must still pass through the lens before reaching the analog or digital sensor. For many applications, it suffices to treat the lens as an ideal pinhole that simply projects all rays through a common center of projection the thin lens composed of a single piece of glass with very low, equal curvature on both sides. According to the lens law

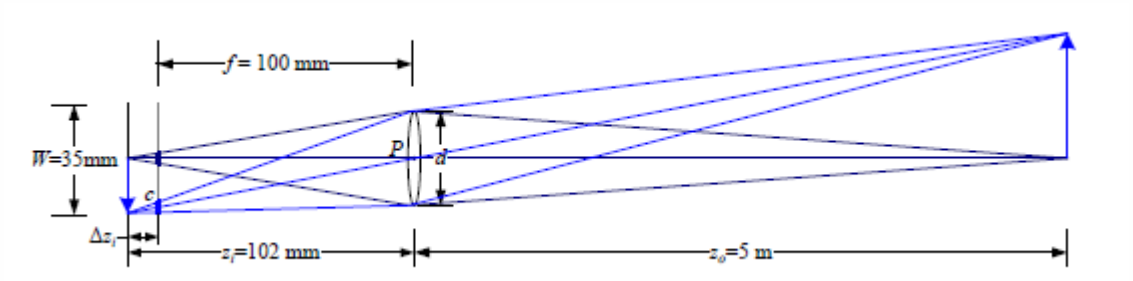


Figure 2.19 *A thin lens of focal length f focuses the light from a plane at a distance z_o in front of the lens onto a plane at a distance z_i behind the lens, where $\frac{1}{z_o} + \frac{1}{z_i} = \frac{1}{f}$. If the focal plane (vertical gray line next to c) is moved forward, the images are no longer 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 zo and the distance behind the lens at which a focused image is formed zi can be expressed as where f is called the focal length of the lens. If we let zo, i.e., we adjust the lens (move the image plane) so that objects at infinity are in focus, we get zi = f, which is why we can think of a lens of focal length f as being equivalent (to a first approximation) to a pinhole at a distance f from the focal plane The allowable depth variation in the scene that limits the circle of confusion to an acceptable

$$\frac{1}{z_o} + \frac{1}{z_i} = \frac{1}{f} \ ,$$

number is commonly called the depth of field and is a function of both the focus distance and the aperture, as shown diagrammatically by many lens markings.

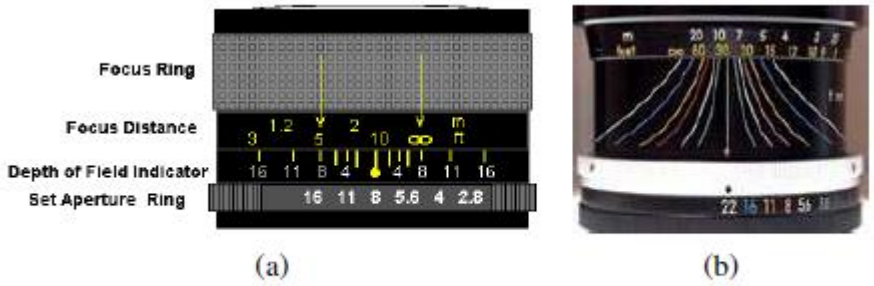


Figure 2.20 *Regular and zoom lens depth of field indicators.*

the focal plane is moved away from its proper in-focus setting of zi (e.g., by twisting the focus ring on the lens), objects at Zo are no longer in focus.

The amount of misfocus is measured by the circle of confusion c (shown as short thick blue line segments on the gray plane). The equation for the circle of confusion can be derived using similar triangles; it depends on the distance of travel in the focal plane ΔZi relative to the original focus distance Zzi and the diameter of the aperture d.

Vignetting

The tendency:y for the brightness of the image to fall off towards the edge of the image.
Two kinds of phenomena usually contribute to this effect.
The first is called natural vignetting and is due to the foreshortening in the object surface, projected pixel, and lens aperture,
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 cosα
The amount of light reaching the lens is also subject to the usual $1/r^2$ fall-off; in this case, the distance in this case, the distance ro = Zo/ cos α. The actual area of the aperture through which the light passes is foreshortened by an additional factor cos α, i.e., the aperture as seen from point *O* is an ellipse of dimensions d*d cos α. 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 is proportional to

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

The digital camera:

After starting from one or more light sources, reflecting off one or more surfaces in the world, and passing through the camera's optics (lenses), light finally reaches the imaging sensor. Light falling on an imaging sensor is usually picked up by an active sensing area, integrated for the duration of the exposure and then passed to a set of sense amplifiers. The two main kinds of sensor used in digital still and video cameras today are charge-coupled device (CCD) and complementary metal oxide on silicon (CMOS).

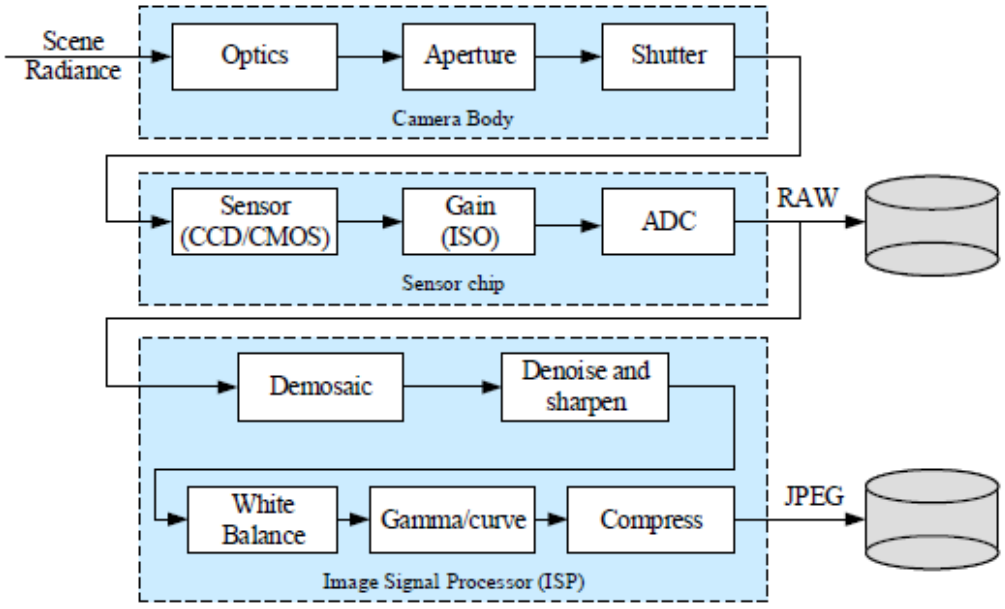


Figure 2.23 *Image sensing pipeline, showing the various sources of noise as well as typical digital post-processing steps.*

In a CCD, photons are accumulated in each active well during the exposure time. Then, in a transfer phase, the charges are transferred from well to well in a kind of "bucket brigade" until they are deposited at the sense amplifiers, which amplify the signal and pass it to an analog-to-digital

converter (ADC).13 Older CCD sensors were prone to blooming, when charges from one over-exposed pixel spilled into adjacent ones, but most newer CCDs have anti-blooming technology

In CMOS, the photons hitting the sensor directly affect the conductivity (or gain) of a photodetector, which can be selectively gated to control exposure duration, and locally amplified before being read out using a multiplexing scheme. Traditionally, CCD sensors outperformed CMOS in quality-sensitive applications, such as digital SLRs, while CMOS was better for low-power applications, but today CMOS is used in most digital cameras.

The main factors affecting the performance of a digital image sensor are the shutter speed, sampling pitch, fill factor, chip size, analog gain, sensor noise, and the resolution (and quality) of the analog-to-digital converter.

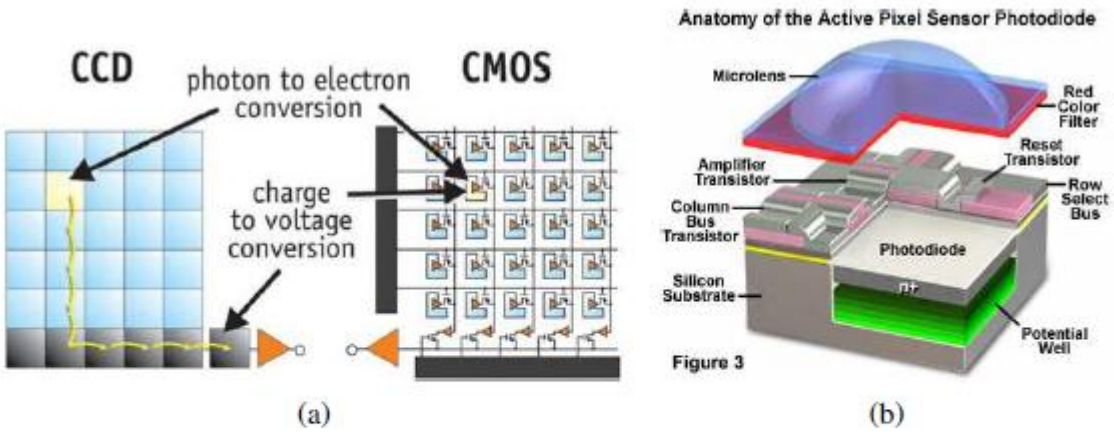


Figure 2.24 *Digital imaging sensors: (a) CCDs move photogenerated charge from pixel to pixel and convert it to voltage at the output node; CMOS imagers convert charge to voltage inside each pixel (Litwiller 2005) © 2005 Photonics Spectra; (b) cutaway diagram of a CMOS pixel sensor, from https://micro.magnet.fsu.edu/primer/digitalimaging/cmosimagesensors.html.*

Shutter speed: The shutter speed (exposure time) directly controls the amount of light reaching the sensor and hence determines if images are under- or over-exposed For dynamic scenes, the shutter speed also determines the amount of motion blur in the resulting picture

Sampling pitch. The sampling pitch is the physical spacing between adjacent sensor cells on the imaging chip A sensor with a smaller sampling pitch has a higher sampling density and hence provides a higher resolution (in terms of pixels) for a given active chip area

Fill factor. The fill factor is the active sensing area size as a fraction of the theoretically available sensing area Higher fill factors are usually preferable, as they result in more light capture and less aliasing.

Chip size. Video and point-and-shoot cameras have traditionally used small chip areas (1/4 - inch to 1/2 -inch sensors), while digital SLR cameras try to come closer to the traditional size of a 35mm film frame. When overall device size is not important, having a larger chip size is preferable, since each sensor cell can be more photo-sensitive.

Analog gain. Before analog-to-digital conversion, the sensed signal is usually boosted by a sense amplifier. In video cameras, the gain on these amplifiers was traditionally controlled by automatic

gain control (AGC) logic, which would adjust these values to obtain a good overall exposure. In newer digital still cameras, the user now has some additional control over this gain through the ISO setting, which is typically expressed in ISO standard units such as 100, 200, or 400

Sensor noise. Throughout the whole sensing process, noise is added from various sources, which may include fixed pattern noise, dark current noise, shot noise, amplifier noise, and quantization noise

ADC resolution. The final step in the analog processing chain occurring within an imaging sensor is the analog to digital conversion (ADC). While a variety of techniques can be used to implement this process, the two quantities of interest are the resolution of this process and its noise level

Digital post-processing. Once the irradiance values arriving at the sensor have been converted to digital bits, most cameras perform a variety of digital signal processing (DSP) operations to enhance the image before compressing and storing the pixel values. These include color filter array (CFA) demosaicing, white point setting, and mapping of the luminance values through a gamma function to increase the perceived dynamic range of the signal.

Newer imaging sensors. The capabilities and compatibility of imaging sensor and related technologies such as depth sensors continue to evolve rapidly. Conferences that track these developments include the IS&T Symposium on Electronic Imaging Science and Technology sponsored by the Society for Imaging Science and Technology and the Image Sensors World blog.

Sampling and aliasing:
The photons arriving at each active cell are integrated and then digitized, if the fill factor on the chip is small and the signal is not otherwise band-limited, visually unpleasing aliasing can occur.

To explore the phenomenon of aliasing, let us first look at a one-dimensional signal in which we have two sine waves, one at a frequency of f = 3/4 and the other at f = 5/4. If we sample these two signals at a frequency of f = 2, we see that they produce the same samples (shown in black), and so we say that they are aliased.

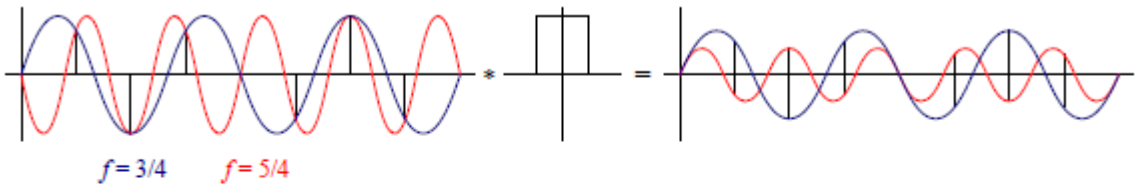


Figure 2.25 *Aliasing of a one-dimensional signal: The blue sine wave at $f = 3/4$ and the red sine wave at $f = 5/4$ have the same digital samples, when sampled at $f = 2$. Even after convolution with a 100% fill factor box filter, the two signals, while no longer of the same magnitude, are still aliased 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.)*

The maximum frequency in a signal is known as the Nyquist frequency and the inverse of the minimum sampling frequency rs = 1/fs is known as the Nyquist rate.

The best way to predict the amount of aliasing that an imaging system will produce is to estimate the point spread function (PSF), which represents the response of a particular pixel sensor to an ideal point light source.

The PSFis a combination (convolution) of the blur induced by the optical system (lens) and the finite integration area of a chip sensor.

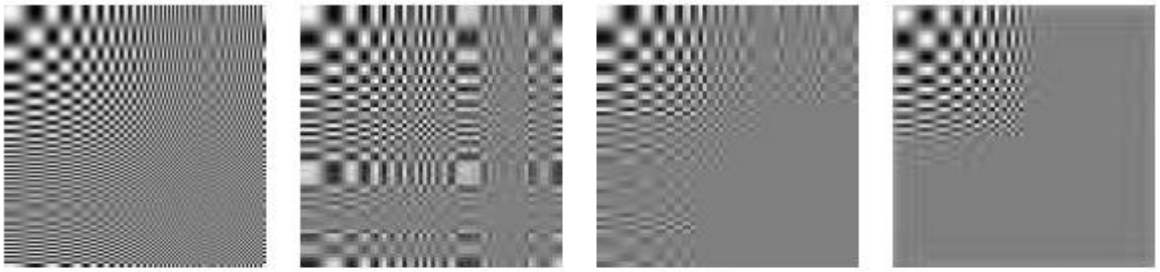


Figure 2.26 *Aliasing of a two-dimensional signal: (a) original full-resolution image; (b) downsampled 4 × with a 25% fill factor box filter; (c) downsampled 4 × with a 100% fill factor box filter; (d) downsampled 4 × with a high-quality 9-tap filter. Notice how the higher frequencies are aliased into visible frequencies with the lower quality filters, while the 9-tap filter completely removes these higher frequencies.*

Color:

When the incoming light hits the imaging sensor, light from different parts of the spectrum is somehow integrated into the discrete red, green, and blue (RGB) color values that we see in a digital image.

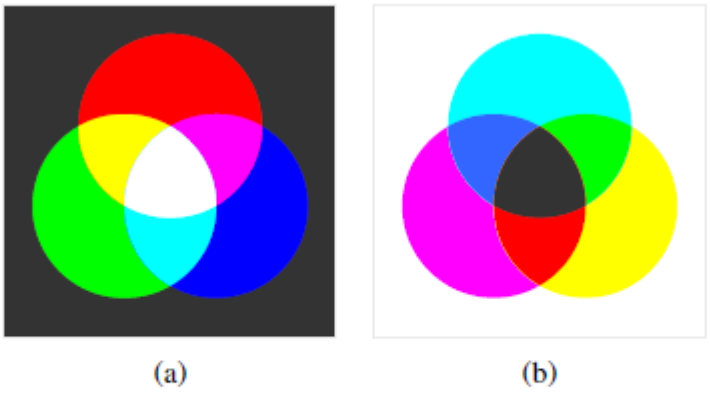


Figure 2.28 *Primary and secondary colors: (a) additive colors red, green, and blue can be mixed to produce cyan, magenta, yellow, and white; (b) subtractive colors cyan, magenta, and yellow can be mixed to produce red, green, blue, and black.*

Color cameras:

The design of RGB video cameras has historically been based around the availability of colored phosphors that go into television sets. When standard-definition color television was invented a mapping was defined between the RGB values that would drive the three color guns in the cathode ray tube (CRT) and the XYZ values that unambiguously define perceived color (this standard was called ITU-R BT.601). With the advent of HDTV and newer monitors, a new standard called ITU-R BT.709 was created, which specifies the XYZ values of each of the color primaries,

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

In practice, each color camera integrates light according to the spectral response function of its red, green, and blue sensors

$$R = \int L(\lambda) S_R(\lambda) d\lambda,$$
$$G = \int L(\lambda) S_G(\lambda) d\lambda,$$
$$B = \int L(\lambda) S_B(\lambda) d\lambda,$$

where L(λ) is the incoming spectrum of light at a given pixel and {S_R(λ); S_G(λ); S_B(λ)} are the red, green, and blue spectral sensitivities of the corresponding sensors.

Colour filter arrays:

While early color TV cameras used three vidicons (tubes) to perform their sensing and later cameras used three separate RGB sensing chips, most of today's digital still and video cameras use a color filter array (CFA)

The most commonly used pattern in color cameras today is the Bayer pattern, which places green filters over half of the sensors (in a checkerboard pattern), and red and blue filters over the remaining ones The reason that there are twice as many green filters as red and blue is because the luminance signal is mostly determined by green values and the visual system is much more sensitive to high-frequency detail in luminance
than in chrominance

The process of interpolating the missing color values so that we have valid RGB values for all the pixels is known as demosaicing

G	R	G	R
B	G	B	G
G	R	G	R
B	G	B	G

(a)

rGb	Rgb	rGb	Rgb
rgB	rGb	rgB	rGb
rGb	Rgb	rGb	Rgb
rgB	rGb	rgB	rGb

(b)

Figure 2.31 *Bayer RGB pattern: (a) color filter array layout; (b) interpolated pixel values, with unknown (guessed) values shown as lower case.*

Gamma:

The relationship between the voltage and the resulting brightness was characterized by a number called gamma (), since the formula was roughly

$$B = V^{\gamma},$$

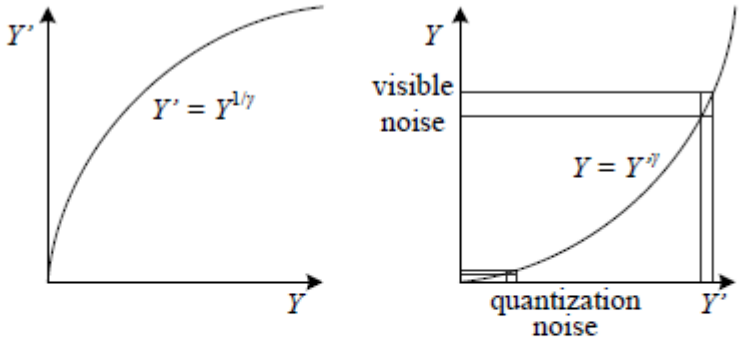


Figure 2.32 *Gamma compression: (a) The relationship between the input signal luminance Y and the transmitted signal Y' is given by $Y' = Y^{1/\gamma}$. (b) At the receiver, the signal Y' is exponentiated by the factor γ, $\hat{Y} = Y'^{\gamma}$. Noise introduced during transmission is squashed in the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

Compression:

The last stage in a camera's processing pipeline is usually some form of image compression All color video and image compression algorithms start by converting the signal into YCbCr (or some closely related variant), so that they can compress the luminance signal with higher fidelity than the chrominance signal. In video, it is common to subsample Cb and Cr by a factor of two horizontally; with still images (JPEG), the subsampling (averaging) occurs both horizontally and vertically.

Once the luminance and chrominance images have been appropriately subsampled and separated into individual images, they are then passed to a block transform stage. The most common technique used here is the discrete cosine transform (DCT), which is a real-valued variant of the discrete Fourier transform (DFT)

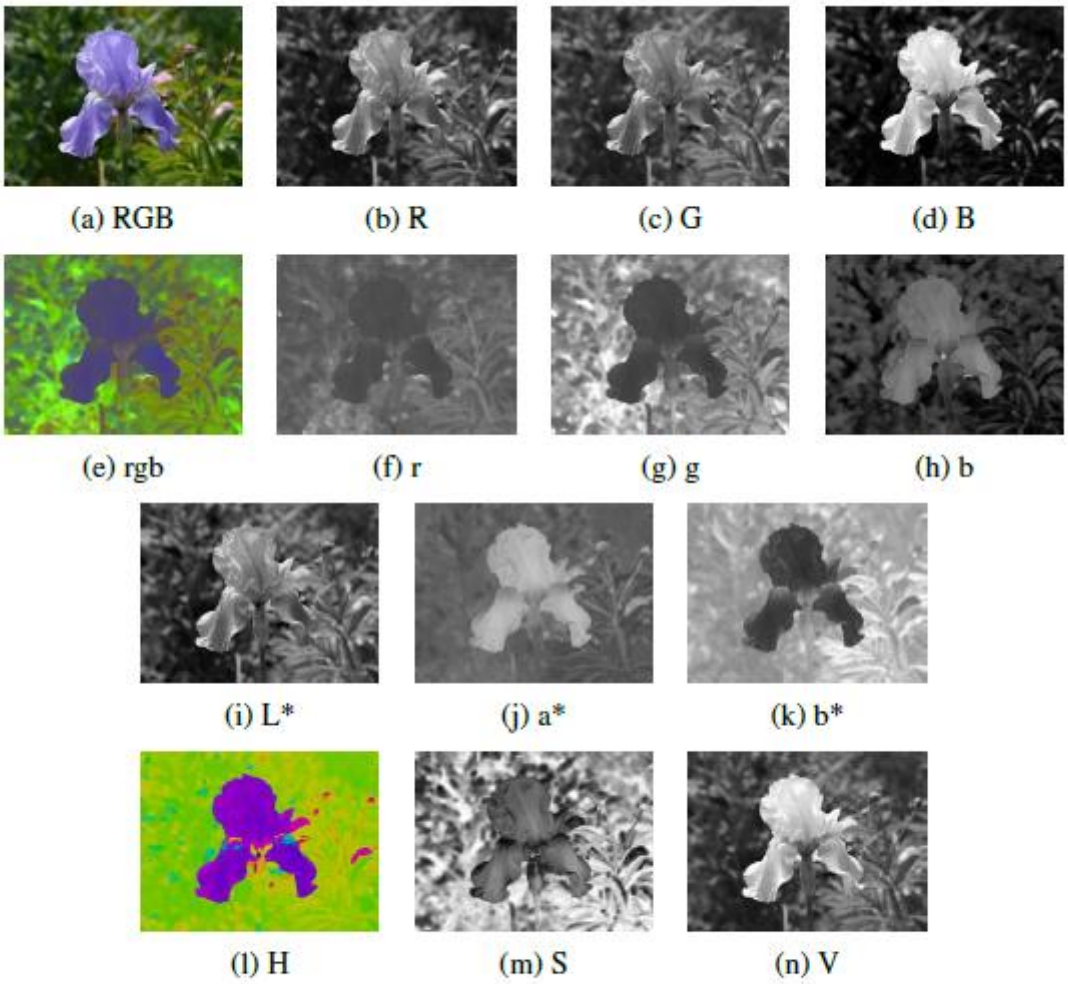


Figure 2.33 *Color space transformations: (a–d) RGB; (e–h) rgb. (i–k) L*a*b*; (l–n) HSV. Note that the rgb, L*a*b*, and HSV values are all re-scaled to fit the dynamic range of the printed page.*

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*,

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

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

$$RMS = \sqrt{MSE}. \tag{2.119}$$

The PSNR is defined as

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

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

Image processing

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.

Pixel transforms:

A general image processing operator is a function that takes one or more input images and produces an output image. In the continuous domain, this can be denoted as

$$g(\mathbf{x}) = h(f(\mathbf{x})) \quad \text{or} \quad g(\mathbf{x}) = h(f_0(\mathbf{x}), \ldots, f_n(\mathbf{x})),$$

where x is in the D-dimensional (usually D = 2 for images) domain of the input and output functions f and g, which operate over some range, which can either be scalar or vector valued, e.g., for color images or 2D motion

For discrete (sampled) images, the domain consists of a finite number of pixel locations, x = (i; j), and we can write

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

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 withstrong connections to the image formation process sensor design and human perception

Color balancing (e.g., to compensate for incandescent lighting) can be performed either by multiplying each channel with a different scale factor or by the more complex process of mapping to XYZ color space, changing the nominal white point, and mapping back to RGB, which can be written down using a linear 3* 3 color twist transform matrix.

Compositing and matting:

The process of extracting the object from the original image is often called matting while the process of inserting it into another image (without visible artifacts) is called compositing

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.*

The intermediate representation used for the foreground object between these two stages is called an alpha-matted color image , In addition to the three color RGB channels, an alpha-matted image contains a fourth alpha channel α (or A) that describes the relative amount of opacity or fractional coverage at each pixel

The opacity is the opposite of the transparency. Pixels within the object are fully opaque (α= 1), while pixels fully outside the object are transparent (α = 0). Pixels on the boundary of the object vary smoothly between these two extremes,

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

This operator attenuates the influence of the background image B by a factor (1-α) and then adds in the color (and opacity) values corresponding to the foreground layer F

Histogram equalization:

the histogram of the individual color channels and luminance values, From this distribution, we can compute relevant statistics, such as the minimum, maximum, and average intensity values. the image will have has both an excess of dark values and light values, but that the mid-range values are largely under-populated. Would it not be better if we could simultaneously brighten some dark values and darken some light values, m histogram equalization, i.e., to find an intensity mapping function f(I) such that the resulting histogram is flat. The trick to finding such a mapping is the same one that people use to generate random samples from a probability density function, which is to first compute the cumulative distribution function

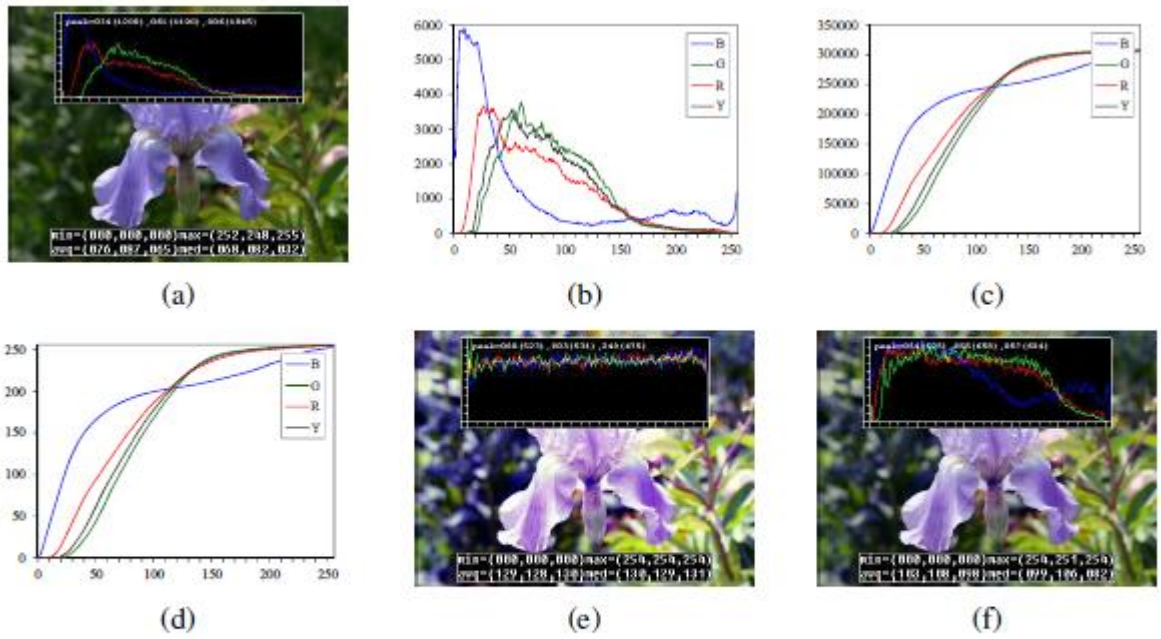


Figure 3.7 *Histogram analysis and equalization: (a) original image; (b) color channel and intensity (luminance) histograms; (c) cumulative distribution functions; (d) equalization (transfer) functions; (e) full histogram equalization; (f) partial histogram equalization.*

Application: Tonal adjustment:

One of the most widely used applications of point-wise image processing operators is the manipulation of contrast or tone in photographs, to make them look either more attractive or more interpretable. You can get a good sense of the range of operations possible by opening up any photo manipulation tool and trying out a variety of contrast, brightness, and color manipulation options,

Linear filtering

Histogram equalization is an example of a neighborhood operator or local operator, which uses a collection of pixel values in the vicinity of a given pixel to determine its final output value neighbourhood operators can be used to filter images to add soft blur, sharpen details, accentuate edges, or remove noise. we look at linear filtering operators, which involve fixed weighted combinations of pixels in small neighborhoods

The most widely used type of neighborhood operator is a linear filter, where an output pixel's value is a weighted sum of pixel values within a small neighborhood N

$$g(i,j) = \sum_{k,l} f(i+k, j+l)h(k,l).$$

The entries in the weight kernel or mask h(k; l) are often called the filter coefficients

45	60	98	127	132	133	137	133
46	65	98	123	126	128	131	133
47	65	96	115	119	123	135	137
47	63	91	107	113	122	138	134
50	59	80	97	110	123	133	134
49	53	68	83	97	113	128	133
50	50	58	70	84	102	116	126
50	50	52	58	69	86	101	120

$f(x,y)$

*

0.1	0.1	0.1
0.1	0.2	0.1
0.1	0.1	0.1

$h(x,y)$

=

69	95	116	125	129	132
68	92	110	120	126	132
66	86	104	114	124	132
62	78	94	108	120	129
57	69	83	98	112	124
53	60	71	85	100	114

$g(x,y)$

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.*

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).*

Padding (border effects):

The astute reader will notice that the correlation produces a result that is smaller than the original image, which may not be desirable in many applications. 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:

- **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.

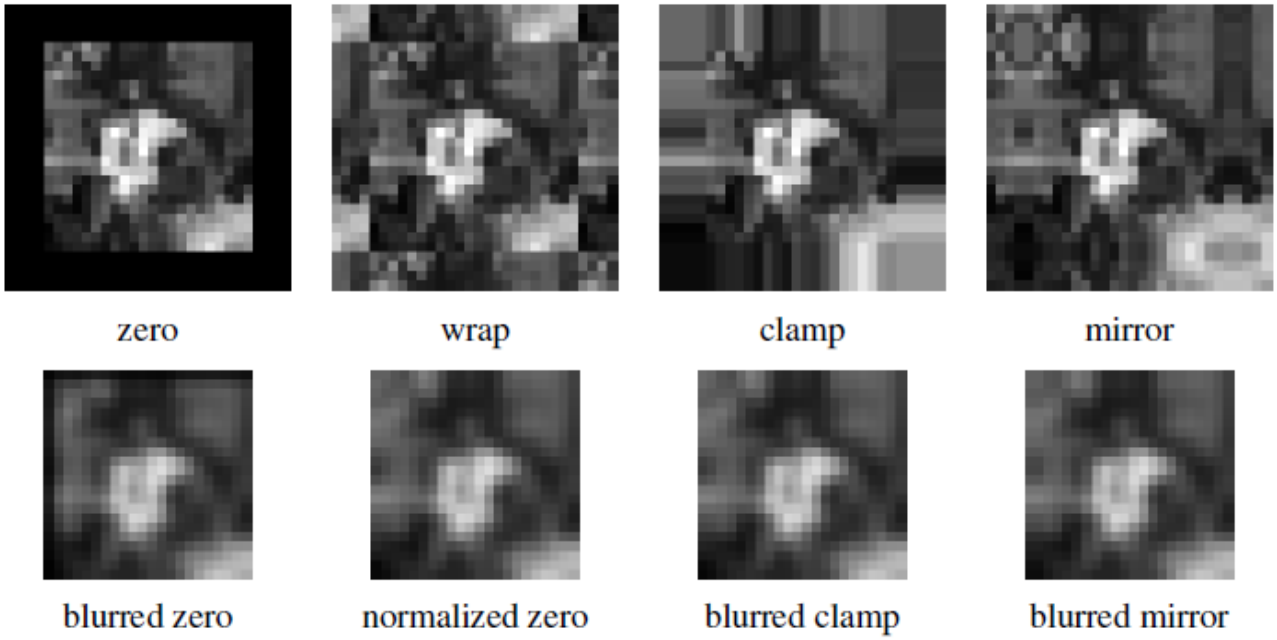


Figure 3.13 *Border padding (top row) and the results of blurring the padded image (bottom row). The normalized zero image is the result of dividing (normalizing) the blurred zero-padded RGBA image by its corresponding soft alpha value.*

Separable filtering:

The process of performing a convolution requires K2 (multiply-add) operations per pixel, where K is the size (width or height) of the convolution kernel, e.g., the box filter.

In many cases, this operation can be significantly sped up by first performing a one-dimensional horizontal convolution followed by a one-dimensional vertical convolution, which requires a total of 2K operations per pixel. A convolution kernel for which this is possible is said to be separable

It is easy to show that the two-dimensional kernel K corresponding to successive convolution with a horizontal kernel h and a vertical kernel v is the outer product of the two kernels,

$$\mathbf{K} = \mathbf{v}\mathbf{h}^T$$

(a) box, $K = 5$ (b) bilinear (c) "Gaussian" (d) Sobel (e) corner

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× and 4×, respectively, and added to a gray offset before display.*

Examples of linear filtering

The simplest filter to implement is the moving average or box filter, which simply averages the pixel values in a K*K window. This is equivalent to convolving the image with a kernel of all ones and then scaling.

For large kernels, a more efficient implementation is to slide a moving window across each scanline (in a separable filter) while adding the newest pixel and subtracting the oldest pixel from the running sum. This is related to the concept of summed area tables,

A smoother image can be obtained by separably convolving the image with a piecewise linear "tent" function (also known as a Bartlett filter , the bilinear kernel, since it is the outer product of two linear (first-order) splines Convolving the linear tent function with itself yields the cubic approximating spline, which is called the "Gaussian" kernel, Note that approximate Gaussian kernels can also be obtained by iterated convolution with box filters

The kernels we just discussed are all examples of blurring (smoothing) or low-pass kernels, since they pass through the lower frequencies while attenuating higher frequencies smoothing kernels can also be used to sharpen images using a process called unsharp masking. Since blurring the image reduces high frequencies, adding some of the

difference between the original and the blurred image makes it sharper

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

Band-pass and steerable filters

The Sobel and corner operators are simple examples of band-pass and oriented filters. More sophisticated kernels can be created by first smoothing the image with a (unit area) Gaussian filter, and then taking the first or second derivatives Such filters are known collectively as band-pass filters, since they filter out both low and high frequencies

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

The (undirected) second derivative of a two-dimensional image

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

is known as the Laplacian operator. Blurring an image with a Gaussian and then taking its Laplacian is equivalent to convolving directly with the Laplacian of Gaussian (LoG) filter, which has certain nice scale-space properties

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

(a) (b) (c)

Figure 3.15 *Second-order steerable filter (Freeman 1992) © 1992 IEEE: (a) original image of Einstein; (b) orientation map computed from the second-order oriented energy; (c) original image with oriented structures enhanced.*

Summed area table (integral image):

If an image is going to be repeatedly convolved with different box filters (and especially filters of different sizes at different locations), you can precompute the summed area table which is just the running sum of all the pixel values from the origin ,

$$s(i,j) = \sum_{k=0}^{i} \sum_{l=0}^{j} f(k,l).$$

, This can be efficiently computed using a recursive (raster-scan) algorithm

$$s(i,j) = s(i-1,j) + s(i,j-1) - s(i-1,j-1) + f(i,j).$$

The image s(i; j) is also often called an integral image and can actually be computed using only two additions per pixel if separate row sums are used

3	2	7	2	3
1	5	1	3	4
5	1	**3**	5	1
4	3	2	1	6
2	4	1	4	8

(a) S = 24

3	5	12	14	17
4	*11*	**19**	24	31
9	**17**	28	38	46
13	24	37	48	62
15	30	44	59	81

(b) s = 28

3	5	12	*14*	17
4	11	19	24	31
9	17	28	38	46
13	24	37	**48**	62
15	30	44	59	81

(c) S = 24

Figure 3.17 *Summed area tables: (a) original image; (b) summed area table; (c) computation of area sum. Each value in the summed area table $s(i,j)$ (red) is computed recursively from its three adjacent (blue) neighbors (3.31). Area sums S (green) are computed by combining the four values at the rectangle corners (purple) (3.32). Positive values are shown in* **bold** *and negative values in italics.*

Recursive filtering:
one whose values depends on previous filter outputs. In the signal processing literature, such filters are known as infinite impulse response (IIR), since the output of the filter to an impulse (single non-zero value) goes on forever. Two-dimensional IIR filters and recursive formulas are sometimes used to compute quantities that involve large area interactions, such as two-dimensional distance functions and connected components