

Segmentation and Classification of Beatboxing Acoustic Voice Tract Variations in MRI through Image Processing Technique

Anurag Sinha Harsh Soni

Department of Information Technology, Research Scholar, Amity University Jharkhand

ABSTRACT

Human beatboxing is a vocal art making use of speech organs to produce vocal drum sounds and imitate musical instruments. Beatbox sound classification is a current challenge that can be used for automatic database annotation and music-information retrieval. In this study, a large-vocabulary humanbeatbox sound recognition system was developed with an adaptation of Kaldi toolbox, a widely-used tool for automatic speech recognition. The corpus consisted of eighty boxemes, which were recorded repeatedly by two beatboxers. The sounds were annotated and transcribed to the system by means of a beatbox specific morphographic writing system (Vocal Grammmatics). The image processing techniques plays vital role on image Acquisition, image pre-processing, Clustering, Segmentation and Classification techniques with different kind of images such as Fruits, Medical, Vehicle and Digital text images etc. In this study the various images to remove unwanted noise and performs enhancement techniques such as contrast limited adaptive histogram equalization, Laplacian and Harr filtering, unsharp masking, sharpening, high boost filtering and color models then the Clustering algorithms are useful for data logically and extract pattern analysis, grouping, decision-making, and machine-learning techniques and Segment the regions using binary, K-means and OTSU segmentation algorithm. It Classifying the images with the help of SVM and K-Nearest Neighbour(KNN) Classifier to produce good results for those images.

KEYWORDS- *Image Acquisition, Image preprocessing, Image enhancement, beatboxing, segmentation*

INTRODUCTION

Speech production involves complex spatiotemporal coordination of several vocal organs in the upper and lower airways. Modalities to study speech production include real-time MRI (RT-MRI), electromagnetic articulography (EMA), electropalatography, ultrasound, and X-ray, or videofluoroscopy (1). In comparison to alternate modalities, RT-MRI provides distinct advantages in terms of (a) noninvasiveness, as opposed to X-rays, videofluoroscopy, and (b) ability to image in arbitrary planes and visualize deep structures (e.g., epiglottis, glottis), which are not possible with ultrasound and EMA. Applications of RT-MRI in speech science and vocal production research are numerous; these include addressing open questions pertaining to understanding the goals of language production, language timing, speech errors, and other topics in phonetics and phonology (1–8) as well as vocal production of song (9,10). It also has the potential to manage and inform treatment plans in several clinical applications, such as clinical assessment of velopharyngeal insufficiency (11–13), cleft palate repair and management (14,15), and surgical planning and post-treatment functional evaluation of speech and swallowing (16,17), in head and neck cancer. The rates of movements of articulators are highly dependent on the speech task and the subject's speaking style (3,18,19). For example, in the articulation of sustained sounds, such as during singing, the spatial position of the articulators change on the order of seconds, whereas in tasks involving flaps/trills and production of consonant clusters, the motion of articulators occur at a much faster rate on the order of a few milliseconds (also see Figure 1) (3).

Whereas modalities such as EMA can operate up to a time resolution of 1 ms, the imaging speed of RT-MRI is restricted by challenges posed as a result of device physics. Several schemes have been proposed to improve the imaging speed of RT-MRI for speech studies. These can be classified as on-the-fly or off-line schemes (3). On-the-fly schemes are referred to those that allow for immediate visualization of the reconstructed images (with latency less than 500 ms), whereas off-line schemes are referred to those where the reconstructions are implemented off-line. Scott et al (20) utilized on-the-fly imaging with Cartesian trajectories and demonstrated temporal resolutions between 111 and 50 ms at spatial resolution of 1.6–2.7 mm² for velopharyngeal closure. Other investigators (1,21,22) utilized short spiral readouts to acquire images at a native time resolution of 54–78 ms, and a spatial resolution of 3.0–2.4 mm², and visualize at 24 frames/sec using view sharing. View sharing was also used with radial imaging in (23,24). Freitas et al (25) compares Cartesian, radial, and spiral trajectories with view sharing and demonstrated spirals to provide the best compromise in terms acquisition efficiency, motion robustness, and signal to noise (SNR). Iterative constrained reconstruction schemes have shown to enable greater acceleration factors. Utilizing radial trajectories, Niebergall et al (23) proposed nonlinear temporal regularized reconstruction to enable a temporal resolution of 33 ms, and a spatial resolution of 1.5 mm², and studied a variety of tasks (vowel, consonant sounds, and coarticulations events). More recently, Iltis et al (26) demonstrated an on-the-fly implementation of the iterative reconstruction by exploiting parallelization within the reconstruction along with efficient use of graphical processing units. This work used a 64-channel brain coil and demonstrated 100 frames/sec at 1.5 mm².

Burdumy et al (24) applied off-line spatial total-variation regularization with radial trajectories to provide spatial resolution of 1.8 mm², and a native time resolution of 55 ms, and analyzed morphometric measurements of the vocal tract. Fu et al (27) utilized off-line reconstruction with the partial separable (PS) model (28,29), along with spatial-spectral sparsity constraint (30), and demonstrated a frame rate of 102 frames/sec at a spatial resolution of 2.2 mm². The low-rank (by the PS model) constraint is essentially a data-driven retrospective binning technique (31) and was fully exploited in Fu et al (27) by utilizing repeated speech utterances. Low-rank constraints are unlikely to apply to short speech utterances with no repetitions, or stimuli involving infrequent distinct movements such as swallowing. In this work, we developed a MRI-based system for dynamic study of vocal tract shaping during speech production to provide high spatiotemporal resolutions. We propose a system that utilizes (a) a novel eight-channel custom upper airway coil, which has improved sensitivity in upper airway regions of interest (ROIs), (b) a flexible slice selective spiral spoiled gradient echo acquisition with golden angle time interleaving, (c) on-the-fly view-sharing reconstruction, (d) off-line temporal finite difference constrained reconstruction, (e) simultaneous audio acquisition, and off-line temporal alignment of noise-cancelled audio with the reconstructions. The innovation of our system lies in the synergistic combination of the above components. We chose custom upper airway coil design for its ability to provide superior SNR across all articulators of interest. This is advantageous because it can provide an important boost in SNR while operating at 1.5 Tesla (T). Its combination with spirals is complementary, because it enables improved SNR at low fields, low off-resonance-related spiral artifacts, and high sampling efficiency. Multishot short spirals (readout length of 2.4 ms) are used to reduce off-resonance artifacts. Our rationale of choosing spirals is that they have shown to provide improved motion robustness and efficiency over alternate trajectories. Temporal finite difference constraint was chosen because it exploits redundancy based on the prior that the desired information is contained in the moving edges, which directly fits to the end goal assessment of dynamics of air-tissue interfaces. Also, because it exploits similarities among neighboring time frames, it is applicable to a wide variety of speech tasks and does not impose restrictions on the imaging task. We present several examples with the proposed system for rapid RT-MRI of speech, including visualization of interleaved consonant and vowel sounds, fluent speech sentences that contain consonant clusters, as

well as beat-boxing sounds that involve rapid coordination between various vocal articulators, on three healthy volunteers, and one tongue cancer patient referred to glossectomy.

Beatboxing is a musical art form in which performers use their vocal tract to create percussive sounds. Sometimes individual beatboxers perform as a part of an ensemble, using their vocal tracts to provide beats for other musicians; other times, beatboxers perform alone, where they might sing and beatbox simultaneously or simply make sounds without singing.

Using the mouth, lips, tongue and voice to generate sounds that one might never expect to come from the human body is the specialty of the artists known as beatboxers. Now scientists have used scanners to peer into a beatboxer as he performed his craft to reveal the secrets of this mysterious art. The human voice has long been used to generate percussion effects in many cultures, including North American scat singing, Celtic lilting and diddling, and Chinese kouji performances. In southern Indian classical music, konnakol is the percussive speech of the solkattu rhythmic form. In contemporary pop music, the relatively young vocal art form of beatboxing is an element of hip-hop culture. Until now, the phonetics of these percussion effects were not examined in detail. For instance, it was unknown to what extent beatboxers produced sounds already used within human language.

To learn more about beatboxing, scientists analyzed a 27-year-old male performing in real-time using MRI. This gave researchers "an opportunity to study the sounds people produce in much greater detail than has previously been possible," said Shrikanth Narayanan, a speech and audio engineer at the University of Southern California in Los Angeles. "The overarching goals of our work drive at larger questions related to the nature of sound production and mental processing in human communication, and a study like this is a small part of the larger puzzle.

Beatboxing is a tradition of vocal percussion which originates in 1980s hip-hop, and is closely connected with hip-hop culture. It involves the vocal imitation of drum machines as well as drums and other percussion, and typically also the simultaneous imitation of basslines, melodies, and vocals, to create an illusion of polyphonic music. It may be performed a capella or with amplification. In this report we describe some characteristics of the beatboxing vocal performance style, as relevant for music signal processing and related fields. In particular we focus on aspects of beatboxing which are different from other vocal styles or from spoken language. Beatboxing developed well outside academia, and separate from the vocal styles commonly studied by universities and conservatories, and so there is (to our knowledge) very little scholarly work on the topic, either its history or its current practice. Beatboxing is mentioned in popular histories of the hip-hop movement, although rarely in detail. An undergraduate thesis looks at phonetic aspects of some beatboxing sounds [Lederer, 2005]. Some technical work is inspired by beatboxing to create (e.g.) a voice-controlled drum-machine [Hazan, 2005a,b, Kapur et al., 2004, Sinyor et al., 2005], although these authors don't make explicit whether their work has been developed in contact with practising beatboxers. In the following we describe characteristics of beatboxing as contrasted against better-documented traditions such as popular singing [Soto-Morettini, 2006] or classical singing [Mabry, 2002]. Because of the relative scarcity of literature, many of the observations come from the first author's experiences and observations: both as a participant in beatboxing communities in the UK and online, and during user studies involving beatboxers as part of the first author's PhD study. We describe certain sounds narratively as well as in International Phonetic Alphabet (IPA) notation [International Phonetic Association, 1999] (see also [Fukui, 2003]), which will be demarcated by slashes //. The IPA representation may be approximate, since the notation is not designed to accommodate easily the non-linguistic and "extended technique" sounds we discuss

. 2 Extended vocal technique Perhaps the most fundamental distinction between the sounds produced while beatboxing and those produced during most other vocal traditions arises from beatboxing's primary aim to create convincing impersonations of drum tracks. (Contrast this against vocal percussion traditions such as jazz scat singing or indian bol, in which percussive rhythms are imitated, but there is no aim to disguise the vocal origin of the sounds.) This aim leads beatboxers to do two things: (1) employ a wide palette of vocal techniques to produce the desired timbres; and (2) suppress some of the linguistic cues that would make clear to an audience that the source is a single human voice. The extended vocal techniques used are many and varied, and vary according to the performer. Many techniques are refinements of standard linguistic vowel and consonant sounds, while some involve sounds that are rarely if at all employed in natural languages. We do not aim to describe all common techniques here, but we will discuss some relatively general aspects of vocal technique which have a noticeable effect on the sound produced.

2.1 Non-syllabic patterns The musical sounds which beatboxers imitate may not sound much like conventional vocal utterances. Therefore the vowel-consonant alternation which is typical of most use of voice may not be entirely suitable for producing a close auditory match. Instead, beatboxers learn to produce sounds to match the sound patterns they aim to replicate, attempting to overcome linguistic patternings. Since human listeners are known to use linguistic sound patterns as one cue to understanding a spoken voice [Shannon et al., 1995], it seems likely that avoiding such patterns may help maintain the illusion of non-voice sound. As mentioned above, vocal traditions such as scat or bol do not aim to disguise the vocal origin of the sounds. Hence in those traditions, patterns are often built up using syllable sounds which do not stray far from the performers' languages.

2.2 Use of inhaled sounds In most singing and spoken language, the vast majority of sounds are produced during exhalation. (Many languages do allow a minor linguistic role for inhaled phonation [Ladefoged and Maddieson, 1997]. Some vocal performance traditions feature inhaled sounds, e.g. Inuit throat-singing games [Nattiez, 2008].) A notable characteristic of beatboxing is the widespread use of inhaled sounds. We propose that this has two main motivations. Firstly it enables a continuous flow of sounds, which both allows for continuous drum patterns and also helps maintain the auditory illusion of the sounds being imitated (since the sound and the pause associated with an ordinary intake of breath are avoided). Secondly it allows for the production of certain sounds which cannot be produced equally well during exhaling. A commonly used example is the "inward clap snare" /ɪ l/ 1. Inhaled sounds are most commonly percussive. Although it is possible to phonate while breathing in, the production of pitched notes while inhaling does not seem to be used much at all by beatboxers. Although some sounds may be specifically produced using inward breath, there are many sounds which beatboxers seem often to be able to produce in either direction, such as the "closed hi-hat" sound /t^/ (outward) or /Ö^/ (inward). This allows some degree of independence between the breathing patterns and the rhythm patterns.

2.3 Vocal modes/qualities Laver [1980] provides the classic phonetician's description of the different voice qualities or "phonatory settings" that an individual can produce, including falsetto, creaky voice, harsh voice, breathy voice, and ventricular voice. (The term "modal voice" is also employed, to refer to the most common vocal quality against which these others are to be distinguished.) These qualities may be consciously manipulated by a speaker, may be part of linguistic distinctions between vowels, or may be indicative of vocal pathology. In study of the singing voice, too, different vocal modes are distinguished [Soto-Morettini, 2006], including head voice, chest voice, belt, twangy voice, growl, breathy voice and creaky voice. Note the (incomplete) overlap between the categories used by the two communities. Beatboxers make use of different vocal qualities to produce specific sounds. For example, growl/ventricular voice may be used to produce a bass tone, and falsetto is used as a component of some sounds, e.g. vocal scratch, "synth kick".

BACKGROUND

The image processing techniques plays vital role on image Acquisition, image pre-processing, Clustering, Segmentation and Classification techniques with different kind of images such as Fruits, Medical, Vehicle and Digital text images etc. In this study the various images to remove unwanted noise and performs enhancement techniques such as contrast limited adaptive histogram equalization, Laplacian and Harr filtering, unsharp masking, sharpening, high boost filtering and color models then the Clustering algorithms are useful for data logically and extract pattern analysis, grouping, decision-making, and machine-learning techniques and Segment the regions using binary,

K-means and OTSU segmentation algorithm. It Classifying the images with the help of SVM and K-Nearest Neighbour(KNN) Classifier to produce good results for those images.

Modern digital technology has made it possible to manipulate multi-dimensional signals with systems that range from simple digital circuits to advanced parallel computers. The goal of this manipulation can be divided into three categories:

- Image Processing $image\ in \rightarrow image\ out$
- Image Analysis $image\ in \rightarrow measurements\ out$
- Image Understanding $image\ in \rightarrow high-level\ description\ out$

We will focus on the fundamental concepts of *image processing*. Space does not permit us to make more than a few introductory remarks about *image analysis*. *Image understanding* requires an approach that differs fundamentally from the theme of this book. Further, we will restrict ourselves to two-dimensional (2D) image processing although most of the concepts and techniques that are to be described can be extended easily to three or more dimensions. Readers interested in either greater detail than presented here or in other aspects of image processing are referred to [1-10]

We begin with certain basic definitions. An image defined in the “real world” is considered to be a function of two real variables, for example, $a(x,y)$ with a as the amplitude (e.g. brightness) of the image at the *real* coordinate position (x,y) . An image may be considered to contain sub-images sometimes referred to as *regions-of-interest*, *ROIs*, or simply *regions*. This concept reflects the fact that images frequently contain collections of objects each of which can be the basis for a region. In a sophisticated image processing system it should be possible to apply specific image processing operations to selected regions. Thus one part of an image (region) might be processed to suppress motion blur while another part might be processed to improve color rendition.

The amplitudes of a given image will almost always be either real numbers or integer numbers. The latter is usually a result of a quantization process that converts a continuous range (say, between 0 and 100%) to a discrete number of levels. In certain image-forming processes, however, the signal may involve photon counting which implies that the amplitude would be inherently quantized. In other image forming procedures, such as magnetic resonance imaging, the direct physical measurement yields a complex number in the form of a real magnitude and a real phase. For the remainder of this book we will consider amplitudes as reals or integers unless otherwise indicated.

2. Digital Image Definitions

A digital image $a[m,n]$ described in a 2D discrete space is derived from an analog image $a(x,y)$ in a 2D continuous space through a *sampling* process that is frequently referred to as digitization. The mathematics of that sampling process will be described in Section 5. For now we will look at some basic definitions associated with the digital image. The effect of digitization is shown in Figure 1.

The 2D continuous image $a(x,y)$ is divided into N rows and M columns. The intersection of a row and a column is termed a *pixel*. The value assigned to the integer coordinates $[m,n]$ with $\{m=0,1,2,\dots,M-1\}$ and $\{n=0,1,2,\dots,N-1\}$ is $a[m,n]$. In fact, in most cases $a(x,y)$ – which we might consider to be the physical signal that impinges on the face of a 2D sensor – is actually a function of many variables including depth (z), color (λ), and time (t). Unless otherwise stated, we will consider the case of 2D, monochromatic, static images in this chapter.

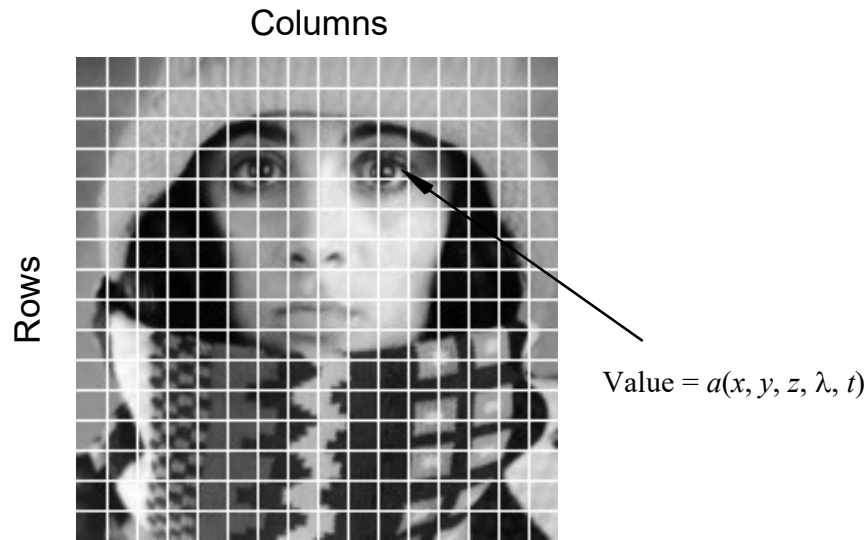


Figure 1: Digitization of a continuous image. The pixel at coordinates $[m=10, n=3]$ has the integer brightness value 110.

The image shown in Figure 1 has been divided into $N = 16$ rows and $M = 16$ columns. The value assigned to every pixel is the average brightness in the pixel rounded to the nearest integer value. The process of representing the amplitude of the 2D signal at a given coordinate as an integer value with L different gray levels is usually referred to as amplitude quantization or simply *quantization*.

COMMON VALUES

There are standard values for the various parameters encountered in digital image processing. These values can be caused by video standards, by algorithmic requirements, or by the desire to keep digital circuitry simple. Table 1 gives some commonly encountered values.

<i>Parameter</i>	<i>Symbol</i>	<i>Typical values</i>
Rows	N	256,512,525,625,1024,1080
Columns	M	256,512,768,1024,1920
Gray Levels	L	2,64,256,1024,4096,16384

Table 1: Common values of digital image parameters

Quite frequently we see cases of $M=N=2^K$ where $\{K = 8,9,10,11,12\}$. This can be motivated by digital circuitry or by the use of certain algorithms such as the (fast) Fourier transform (see Section 3.3).

The number of distinct gray levels is usually a power of 2, that is, $L=2^B$ where B is the number of bits in the binary representation of the brightness levels. When $B>1$ we speak of a *gray-level image*; when $B=1$ we speak of a *binary image*. In a binary image there are just two gray levels which can be referred to, for example, as “black” and “white” or “0” and “1”.

CHARACTERISTICS OF IMAGE OPERATIONS

There is a variety of ways to classify and characterize image operations. The reason for doing so is to understand what type of results we might expect to achieve with a given type of operation or what might be the computational burden associated with a given operation.

Types of operations

The types of operations that can be applied to digital images to transform an input image $a[m,n]$ into an output image $b[m,n]$ (or another representation) can be classified into three categories as shown in Table 2.

Operation	Characterization	Generic Complexity/Pixel
• <i>Point</i>	– the output value at a specific coordinate is dependent only on the input value at that same coordinate.	<i>constant</i>
• <i>Local</i>	– the output value at a specific coordinate is dependent on the input values in the <i>neighborhood</i> of that same coordinate.	P^2
• <i>Global</i>	– the output value at a specific coordinate is dependent on all the values in the input image.	N^2

Table 2: Types of image operations. Image size = $N \times N$; neighborhood size = $P \times P$. Note that the complexity is specified in operations *per pixel*.

This is shown graphically in Figure 2.

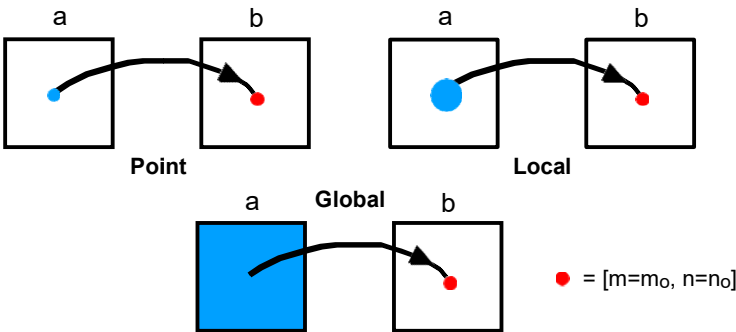


Figure 2: Illustration of various types of image operations

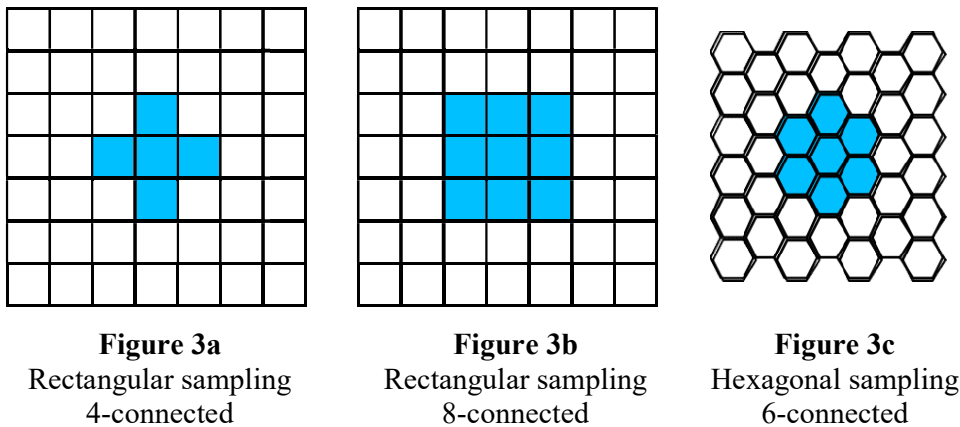
Types of neighborhoods

Neighborhood operations play a key role in modern digital image processing. It is therefore important to understand how images can be sampled and how that relates to the various neighborhoods that can be used to process an image.

- Rectangular sampling – In most cases, images are sampled by laying a rectangular grid over an image as illustrated in Figure 1. This results in the type of sampling shown in Figure 3ab.
- Hexagonal sampling – An alternative sampling scheme is shown in Figure 3c and is termed hexagonal sampling.

Both sampling schemes have been studied extensively [1] and both represent a possible periodic tiling of the continuous image space. We will restrict our attention, however, to only rectangular sampling as it remains, due to hardware and software considerations, the method of choice.

Local operations produce an output pixel value $b[m=m_o,n=n_o]$ based upon the pixel values in the *neighborhood* of $a[m=m_o,n=n_o]$. Some of the most common neighborhoods are the 4-connected neighborhood and the 8-connected neighborhood in the case of rectangular sampling and the 6-connected neighborhood in the case of hexagonal sampling illustrated in Figure 3.



VIDEO PARAMETERS

We do not propose to describe the processing of dynamically changing images in this introduction. It is appropriate—given that many static images are derived from video cameras and frame grabbers—to mention the standards that are associated with the three standard video schemes that are currently in worldwide use – NTSC, PAL, and SECAM. This information is summarized in Table 3.

<i>Property</i>	<i>Standard</i>	NTSC	PAL	SECAM
images / second		29.97	25	25
ms / image		33.37	40.0	40.0
lines / image		525	625	625
(horiz./vert.) = aspect ratio		4:3	4:3	4:3
interlace		2:1	2:1	2:1
μ s / line		63.56	64.00	64.00

Table 3: Standard video parameters

In an interlaced image the odd numbered lines (1,3,5,...) are scanned in half of the allotted time (e.g. 20 ms in PAL) and the even numbered lines (2,4,6,...) are scanned in the remaining half. The image display must be coordinated with this scanning format. (See Section 8.2.) The reason for interlacing the scan lines of a video image is to reduce the perception of flicker in a displayed image. If one is planning to use images that have been scanned from an interlaced video source, it is important to know if the two half-images have been appropriately “shuffled” by the digitization hardware or if that should be implemented in software. Further, the analysis of moving objects requires special care with interlaced video to avoid “zigzag” edges.

The number of rows (N) from a video source generally corresponds one-to-one with lines in the video image. The number of columns, however, depends on the nature of the electronics that is used to digitize the image. Different frame grabbers for the same video camera might produce $M = 384, 512$, or 768 columns (pixels) per line.

3. Tools

Certain tools are central to the processing of digital images. These include mathematical tools such as *convolution*, *Fourier analysis*, and *statistical* descriptions, and manipulative tools such as *chain codes* and *run codes*. We will present these tools without any specific motivation. The motivation will follow in later sections.

CONVOLUTION

There are several possible notations to indicate the convolution of two (multi-dimensional) signals to produce an output signal. The most common are:**g**

$$c = a \otimes b = a * b \quad (1)$$

We shall use the first form, $c = a \otimes b$, with the following formal definitions.

In 2D continuous space:

$$c(x, y) = a(x, y) \otimes b(x, y) = \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} a(\chi, \zeta) b(x - \chi, y - \zeta) d\chi d\zeta \quad (2)$$

In 2D discrete space:

$$c[m, n] = a[m, n] \otimes b[m, n] = \sum_{j=-\infty}^{+\infty} \sum_{k=-\infty}^{+\infty} a[j, k] b[m - j, n - k] \quad (3)$$

PROPERTIES OF CONVOLUTION

There are a number of important mathematical properties associated with convolution.

- Convolution is *commutative*.

$$c = a \otimes b = b \otimes a \quad (4)$$

- Convolution is *associative*.

$$c = a \otimes (b \otimes d) = (a \otimes b) \otimes d = a \otimes b \otimes d \quad (5)$$

- Convolution is *distributive*.

$$c = a \otimes (b + d) = (a \otimes b) + (a \otimes d) \quad (6)$$

where a , b , c , and d are all images, either continuous or discrete.

FOURIER TRANSFORMS

The Fourier transform produces another representation of a signal, specifically a representation as a weighted sum of complex exponentials. Because of Euler's formula:

$$e^{jq} = \cos(q) + j \sin(q) \quad (7)$$

where $j^2 = -1$, we can say that the Fourier transform produces a representation of a (2D) signal as a weighted sum of sines and cosines. The defining formulas for

the forward Fourier and the inverse Fourier transforms are as follows. Given an image a and its Fourier transform A , then the forward transform goes from the spatial domain (either continuous or discrete) to the frequency domain which is always continuous.

$$\text{Forward} - \quad A = \mathbf{F}\{a\} \quad (8)$$

The inverse Fourier transform goes from the frequency domain back to the spatial domain.

$$\text{Inverse} - \quad a = \mathbf{F}^{-1}\{A\} \quad (9)$$

The Fourier transform is a unique and invertible operation so that:

$$a = \mathbf{F}^{-1}\{\mathbf{F}\{a\}\} \quad \text{and} \quad A = \mathbf{F}\{\mathbf{F}^{-1}\{A\}\} \quad (10)$$

The specific formulas for transforming back and forth between the spatial domain and the frequency domain are given below.

In 2D continuous space:

$$\text{Forward} - \quad A(u, v) = \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} a(x, y) e^{-j(ux+vy)} dx dy \quad (11)$$

$$\text{Inverse} - \quad a(x, y) = \frac{1}{4\pi^2} \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} A(u, v) e^{+j(ux+vy)} du dv \quad (12)$$

In 2D discrete space:

$$\text{Forward} - \quad A(\wedge, \Psi) = \sum_{m=-\infty}^{+\infty} \sum_{n=-\infty}^{+\infty} a[m, n] e^{-j(\wedge m + \Psi n)} \quad (13)$$

$$\text{Inverse} - \quad a[m, n] = \frac{1}{4\pi^2} \int_{-\pi}^{+\pi} \int_{-\pi}^{+\pi} A(\wedge, \Psi) e^{+j(\wedge m + \Psi n)} d\wedge d\Psi \quad (14)$$

PROPERTIES OF FOURIER TRANSFORMS

There are a variety of properties associated with the Fourier transform and the inverse Fourier transform. The following are some of the most relevant for digital image processing.

- The Fourier transform is, in general, a complex function of the real frequency variables. As such the transform can be written in terms of its magnitude and phase.

$$A(u, v) = |A(u, v)| e^{j\varphi(u, v)} \quad A(\wedge, \Psi) = |A(\wedge, \Psi)| e^{j\varphi(\wedge, \Psi)} \quad (15)$$

- A 2D signal can also be complex and thus written in terms of its magnitude and phase.

$$a(x, y) = |a(x, y)| e^{j\vartheta(x, y)} \quad a[m, n] = |a[m, n]| e^{j\vartheta[m, n]} \quad (16)$$

- If a 2D signal is real, then the Fourier transform has certain symmetries.

$$A(u, v) = A^*(-u, -v) \quad A(\wedge, \Psi) = A^*(-\wedge, -\Psi) \quad (17)$$

The symbol (*) indicates complex conjugation. For real signals eq. (17) leads directly to:

$$\begin{aligned} |A(u, v)| &= |A(-u, -v)| & \varphi(u, v) &= -\varphi(-u, -v) \\ |A(\wedge, \Psi)| &= |A(-\wedge, -\Psi)| & \varphi(\wedge, \Psi) &= -\varphi(-\wedge, -\Psi) \end{aligned} \quad (18)$$

- If a 2D signal is real and even, then the Fourier transform is real and even.

$$A(u, v) = A(-u, -v) \quad A(\wedge, \Psi) = A(-\wedge, -\Psi) \quad (19)$$

- The Fourier and the inverse Fourier transforms are linear operations.

$$\begin{aligned} \mathbf{F}\{w_1 a + w_2 b\} &= \mathbf{F}\{w_1 a\} + \mathbf{F}\{w_2 b\} = w_1 A + w_2 B \\ \mathbf{F}^{-1}\{w_1 A + w_2 B\} &= \mathbf{F}^{-1}\{w_1 A\} + \mathbf{F}^{-1}\{w_2 B\} = w_1 a + w_2 b \end{aligned} \quad (20)$$

where a and b are 2D signals (images) and w_1 and w_2 are arbitrary, complex constants.

- The Fourier transform in discrete space, $A(\wedge, \Psi)$, is periodic in both \wedge and Ψ . Both periods are 2π .

$$A(\wedge + 2\pi j, \Psi + 2\pi k) = A(\wedge, \Psi) \quad j, k \text{ integers} \quad (21)$$

- The energy, E , in a signal can be measured either in the spatial domain or the frequency domain. For a signal with finite energy:

Parseval's theorem (2D continuous space):

$$E = \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} |a(x, y)|^2 dx dy = \frac{1}{4\pi^2} \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} |A(u, v)|^2 du dv \quad (22)$$

Parseval's theorem (2D discrete space):

$$E = \sum_{m=-\infty}^{+\infty} \sum_{n=-\infty}^{+\infty} |a[m, n]|^2 = \frac{1}{4\pi^2} \int_{-\pi}^{+\pi} \int_{-\pi}^{+\pi} |A(\wedge, \Psi)|^2 d\wedge d\Psi \quad (23)$$

This “signal energy” is not to be confused with the physical energy in the phenomenon that produced the signal. If, for example, the value $a[m, n]$ represents a photon count, then the *physical* energy is proportional to the amplitude, a , and not the square of the amplitude. This is generally the case in video imaging.

- Given three, multi-dimensional signals a , b , and c and their Fourier transforms A , B , and C :

$$c = a \otimes b \quad \xleftrightarrow{F} \quad C = A \cdot B \quad (24)$$

and

$$c = a \cdot b \quad \xleftrightarrow{F} \quad C = \frac{1}{4\pi^2} A \otimes B$$

In words, convolution in the spatial domain is equivalent to multiplication in the Fourier (frequency) domain and vice-versa. This is a central result which provides not only a methodology for the implementation of a convolution but also insight into how two signals interact with each other—under convolution—to produce a third signal. We shall make extensive use of this result later.

- If a two-dimensional signal $a(x, y)$ is scaled in its spatial coordinates then:

$$\begin{aligned} \text{If } a(x, y) &\rightarrow a(M_x \cdot x, M_y \cdot y) \\ \text{Then } A(u, v) &\rightarrow A \left[\frac{u}{M_x}, \frac{v}{M_y} \right] \cdot |M_x \cdot M_y| \end{aligned} \quad (25)$$

- If a two-dimensional signal $a(x,y)$ has Fourier spectrum $A(u,v)$ then:

$$A(u=0, v=0) = \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} a(x, y) dx dy$$

$$a(x=0, y=0) = \frac{1}{4\pi^2} \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} A(u, v) dx dy$$
(26)

- If a two-dimensional signal $a(x,y)$ has Fourier spectrum $A(u,v)$ then:

$$\begin{aligned} \frac{\partial a(x, y)}{\partial x} &\xleftrightarrow{F} juA(u, v) & \frac{\partial a(x, y)}{\partial y} &\xleftrightarrow{F} jvA(u, v) \\ \frac{\partial^2 a(x, y)}{\partial x^2} &\xleftrightarrow{F} -u^2 A(u, v) & \frac{\partial^2 a(x, y)}{\partial y^2} &\xleftrightarrow{F} -v^2 A(u, v) \end{aligned}$$
(27)

Importance of phase and magnitude

Equation (15) indicates that the Fourier transform of an image can be complex. This is illustrated below in Figures 4a-c. Figure 4a shows the original image $a[m,n]$, Figure 4b the magnitude in a scaled form as $\log(|A(\wedge, \Psi)|)$, and Figure 4c the phase $\varphi(\wedge, \Psi)$.



Figure 4a

Original

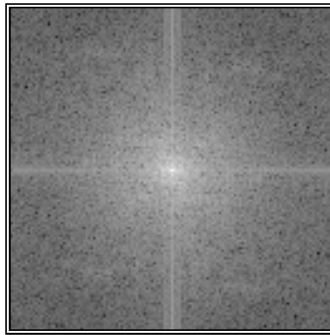


Figure 4b

$\log(|A(\wedge, \Psi)|)$

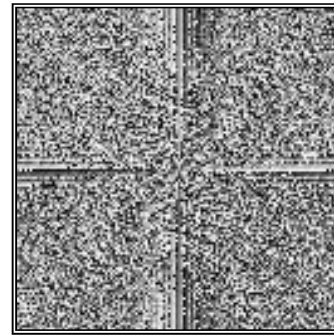
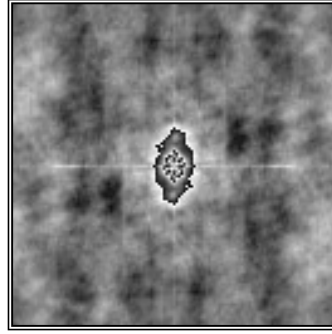


Figure 4c

$\varphi(\wedge, \Psi)$

Both the magnitude and the phase functions are necessary for the complete reconstruction of an image from its Fourier transform. Figure 5a shows what happens when Figure 4a is restored solely on the basis of the magnitude information and Figure 5b shows what happens when Figure 4a is restored solely on the basis of the phase information.

**Figure 5a**

$$\varphi(\wedge, \Psi) = 0$$

**Figure 5b**

$$|A(\wedge, \Psi)| = \text{constant}$$

Neither the magnitude information nor the phase information is sufficient to restore the image. The magnitude-only image (Figure 5a) is unrecognizable and has severe dynamic range problems. The phase-only image (Figure 5b) is barely recognizable, that is, severely degraded in quality.

Circularly symmetric signals

An arbitrary 2D signal $a(x, y)$ can always be written in a polar coordinate system as $a(r, \theta)$. When the 2D signal exhibits a circular symmetry this means that:

$$a(x, y) = a(r, \theta) = a(r) \quad (28)$$

where $r^2 = x^2 + y^2$ and $\tan \theta = y/x$. As a number of physical systems such as lenses exhibit circular symmetry, it is useful to be able to compute an appropriate Fourier representation.

The Fourier transform $A(u, v)$ can be written in polar coordinates $A(q, \xi)$ and then, for a circularly symmetric signal, rewritten as a *Hankel transform*:

$$A(u, v) = \mathbf{F} \{ a(x, y) \} = 2\pi \int_0^{\infty} a(r) J_0(rq) r dr = A(q) \quad (29)$$

where $q^2 = u^2 + v^2$ and $\tan \xi = v/u$ and $J_0(\bullet)$ is a Bessel function of the first kind of order zero.

The inverse *Hankel transform* is given by:

$$a(r) = \frac{1}{2\pi} \int_0^{\infty} A(q) J_0(rq) q dq \quad (30)$$

The Fourier transform of a circularly symmetric 2D signal is a function of only the radial frequency, q . The dependence on the angular frequency, ξ , has vanished. Further, if $a(x,y) = a(r)$ is real, then it is automatically even due to the circular symmetry. According to equation (19), $A(q)$ will then be real and even.

Examples of 2D signals and transforms

Table 4 shows some basic and useful signals and their 2D Fourier transforms. In using the table entries in the remainder of this chapter we will refer to a spatial domain term as the *point spread function (PSF)* or the *2D impulse response* and its Fourier transforms as the *optical transfer function (OTF)* or simply *transfer function*. Two standard signals used in this table are $u(\bullet)$, the unit step function, and $J_1(\bullet)$, the Bessel function of the first kind. Circularly symmetric signals are treated as functions of r as in eq. (28).

STATISTICS

In image processing it is quite common to use simple statistical descriptions of images and sub-images. The notion of a statistic is intimately connected to the concept of a probability distribution, generally the distribution of signal amplitudes. For a given region—which could conceivably be an entire image—we can define the probability *distribution* function of the brightnesses in that region and the probability *density* function of the brightnesses in that region. We will assume in the discussion that follows that we are dealing with a digitized image $a[m,n]$.

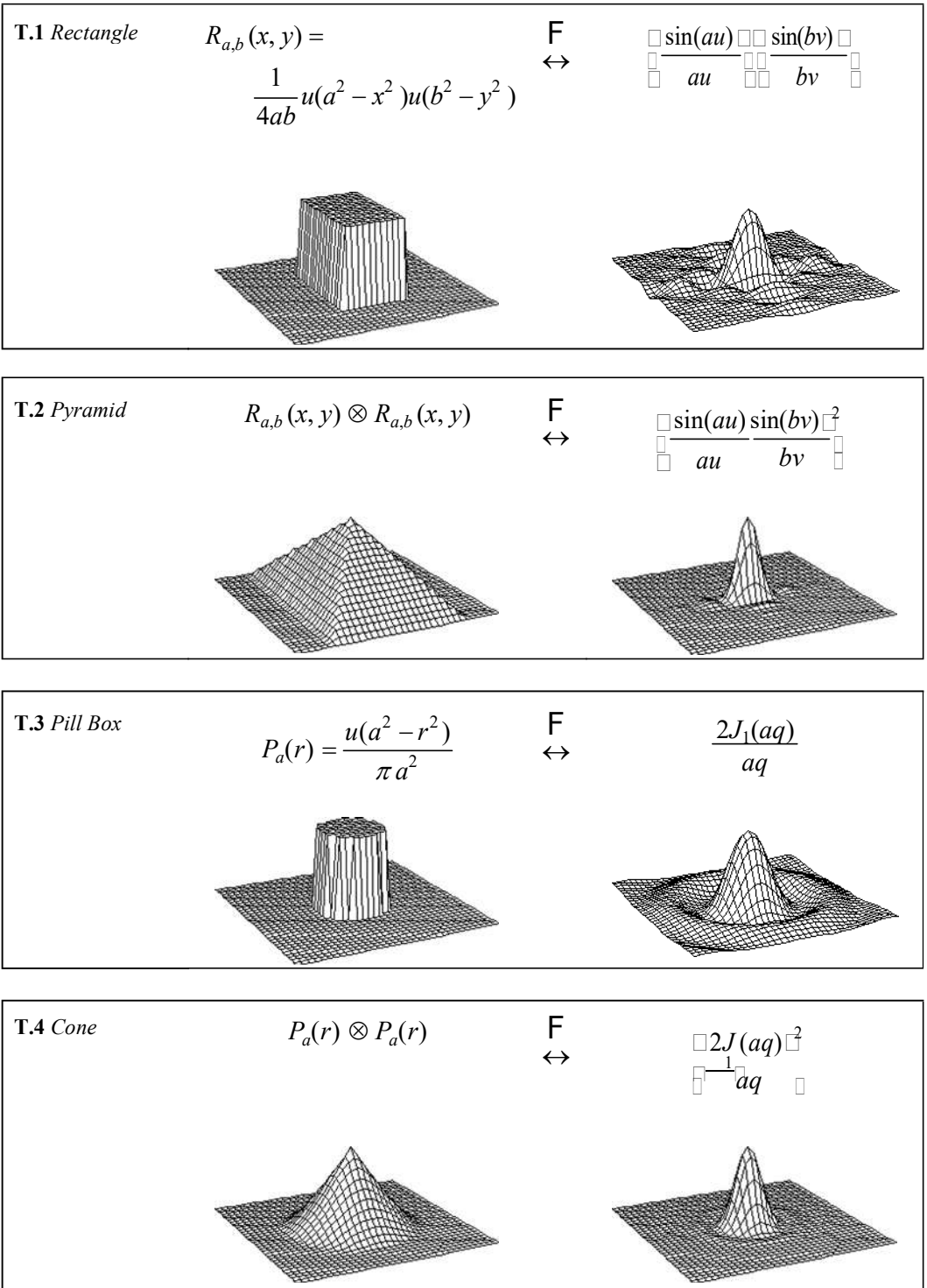
Probability distribution function of the brightnesses

The probability distribution function, $P(a)$, is the probability that a brightness chosen from the region is less than or equal to a given brightness value a . As a increases from $-\infty$ to $+\infty$, $P(a)$ increases from 0 to 1. $P(a)$ is monotonic, non-decreasing in a and thus $dP/da \geq 0$.

Probability density function of the brightnesses

The probability that a brightness in a region falls between a and $a+\Delta a$, given the probability distribution function $P(a)$, can be expressed as $p(a)\Delta a$ where $p(a)$ is the probability density function:

$$p(a)\Delta a = \frac{dP(a)}{da} \Delta a \quad (31)$$



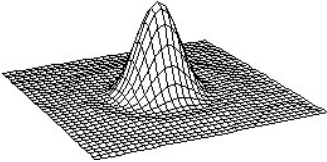
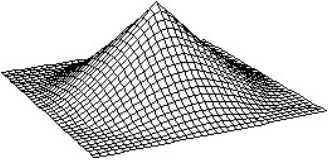
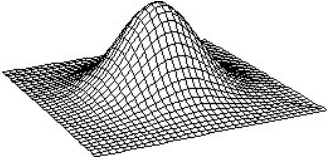
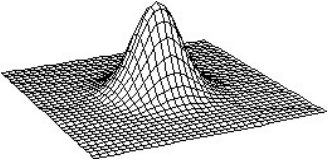
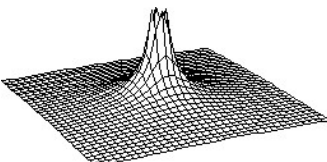
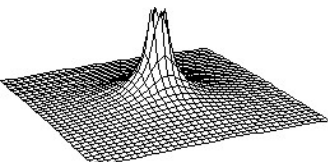
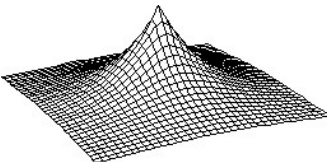
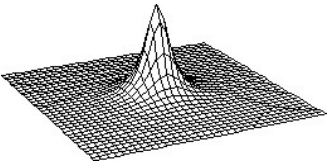
T.5 <i>Airy PSF</i>	$PSF(r) = \frac{1}{\pi} \frac{J_1^2(\frac{1}{2} q r)}{r}$	$\overset{\text{F}}{\leftrightarrow} \frac{2}{\pi} \cos^{-1} \frac{q}{q_c} - \frac{q}{q_c} \sqrt{1 - \frac{q^2}{q_c^2}} u\left(\frac{2}{q_c} q^2\right)$ with $q_c = 2\pi NA/\lambda$
		
T.6 <i>Gaussian</i>	$g_{2D}(r, \sigma) = \frac{1}{2\pi\sigma^2} \exp\left(-\frac{r^2}{2\sigma^2}\right)$	$\overset{\text{F}}{\leftrightarrow} G_{2D}(q, \sigma) = \exp\left(-q^2 \sigma^2\right)$
		
T.7 <i>Peak</i>	$\frac{1}{r}$	$\overset{\text{F}}{\leftrightarrow} \frac{2\pi}{q}$
		
T.8 <i>Exponential Decay</i>	e^{-ar}	$\overset{\text{F}}{\leftrightarrow} \frac{2\pi a}{(a^2 + q^2)^{3/2}}$
		

Table 4: 2D Images and their Fourier Transforms

Because of the monotonic, non-decreasing character of $P(a)$ we have that:

$$p(a) \geq 0 \quad \text{and} \quad \int_{-\infty}^{+\infty} p(a) da = 1 \quad (32)$$

For an image with quantized (integer) brightness amplitudes, the interpretation of $\otimes a$ is the width of a brightness interval. We assume constant width intervals. The brightness probability *density* function is frequently estimated by counting the number of times that each brightness occurs in the region to generate a *histogram*, $h[a]$. The histogram can then be normalized so that the total area under the histogram is 1 (eq. (32)). Said another way, the $p[a]$ for a region is the normalized count of the number of pixels, Λ , in a region that have quantized brightness a :

$$p[a] = \frac{1}{\Lambda} h[a] \quad \text{with} \quad \Lambda = \sum_a h[a] \quad (33)$$

The brightness probability *distribution* function for the image shown in Figure 4a is shown in Figure 6a. The (unnormalized) brightness histogram of Figure 4a which is proportional to the estimated brightness probability density function is shown in Figure 6b. The height in this histogram corresponds to the number of pixels with a given brightness.

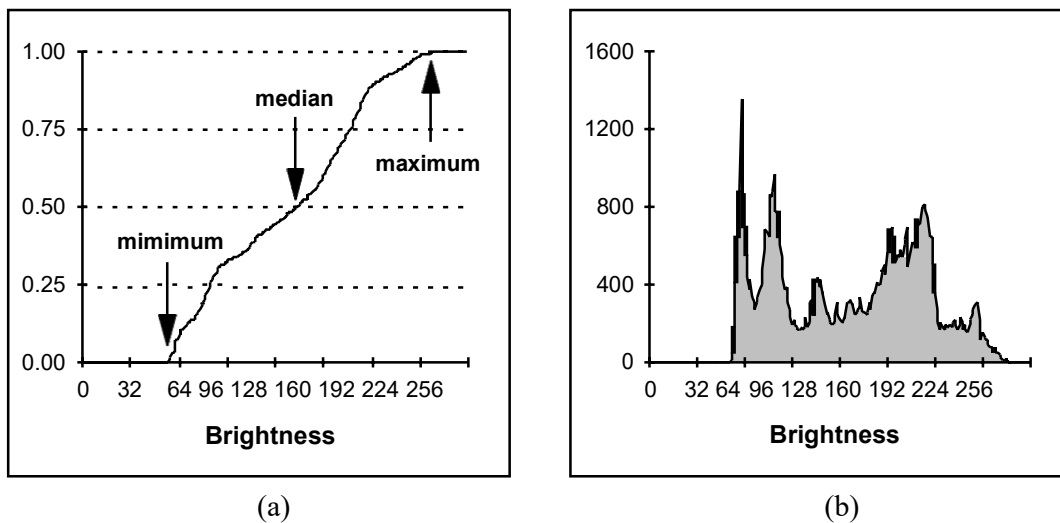


Figure 6: (a) Brightness distribution function of Figure 4a with *minimum*, *median*, and *maximum* indicated. See text for explanation. (b) Brightness histogram of Figure 4a.

Both the distribution function and the histogram as measured from a region are a statistical description of that region. It must be emphasized that both $P[a]$ and $p[a]$ should be viewed as *estimates* of true distributions when they are computed from

a specific region. That is, we view an image and a specific region as one realization of the various random processes involved in the formation of that image and that region. In the same context, the statistics defined below must be viewed as estimates of the underlying parameters.

Average

The average brightness of a region is defined as the *sample mean* of the pixel brightnesses within that region. The average, m_a , of the brightnesses over the Λ pixels within a region (\mathfrak{R}) is given by:

$$m_a = \frac{1}{\Lambda} \sum_{(m,n) \in \mathfrak{R}} a[m,n] \quad (34)$$

Alternatively, we can use a formulation based upon the (unnormalized) brightness histogram, $h(a) = \Lambda \cdot p(a)$, with discrete brightness values a . This gives:

$$m_a = \frac{1}{\Lambda} \sum_a a \cdot h[a] \quad (35)$$

The average brightness, m_a , is an estimate of the mean brightness, μ_a , of the underlying brightness probability distribution.

Standard deviation

The *unbiased estimate* of the standard deviation, s_a , of the brightnesses within a region (\mathfrak{R}) with Λ pixels is called the *sample standard deviation* and is given by:

$$\begin{aligned} s_a &= \sqrt{\frac{1}{\Lambda - 1} \sum_{m,n \in \mathfrak{R}} (a[m,n] - m_a)^2} \\ &= \sqrt{\frac{\sum_{m,n \in \mathfrak{R}} a^2[m,n] - \Lambda m_a^2}{\Lambda - 1}} \end{aligned} \quad (36)$$

Using the histogram formulation gives:

$$s_a = \sqrt{\frac{\sum_a a^2 \cdot h[a] - \Lambda \cdot m_a^2}{\Lambda - 1}} \quad (37)$$

The standard deviation, s_a , is an estimate of σ_a of the underlying brightness probability distribution.

Coefficient-of-variation

The dimensionless *coefficient-of-variation*, CV , is defined as:

$$CV = \frac{s_a}{m_a} \times 100\% \quad (38)$$

Percentiles

The percentile, $p\%$, of an *unquantized* brightness distribution is defined as that value of the brightness a such that:

$$P(a) = p\%$$

or equivalently

$$\int_{-\infty}^a p(\alpha) d\alpha = p\% \quad (39)$$

Three special cases are frequently used in digital image processing.

- 0% the *minimum* value in the region
- 50% the *median* value in the region
- 100% the *maximum* value in the region

All three of these values can be determined from Figure 6a.

Mode

The mode of the distribution is the most frequent brightness value. There is no guarantee that a mode exists or that it is unique.

Signal-to-Noise ratio

The signal-to-noise ratio, SNR , can have several definitions. The noise is characterized by its standard deviation, s_n . The characterization of the signal can differ. If the signal is known to lie between two boundaries, $a_{min} \leq a \leq a_{max}$, then the SNR is defined as:

$$\text{Bounded signal} - \quad SNR = 20 \log_{10} \frac{a_{max} - a_{min}}{s_n} \text{ dB} \quad (40)$$

If the signal is not bounded but has a statistical distribution then two other definitions are known:

$$\begin{array}{l} \text{Stochastic signal} - \\ \text{S \& N inter-dependent} \end{array} \quad SNR = 20 \log_{10} \frac{m_a}{s_n} \text{ dB} \quad (41)$$

$$S \text{ \& } N \text{ independent} \quad SNR = 20 \log_{10} \frac{s_a}{s_n} \text{ dB} \quad (42)$$

where m_a and s_a are defined above.

The various statistics are given in Table 5 for the image and the region shown in Figure 7.



Figure 7

Region is the interior of the circle.

Statistic	Image	ROI
Average	137.7	219.3
Standard Deviation	49.5	4.0
Minimum	56	202
Median	141	220
Maximum	241	226
Mode	62	220
SNR (db)	NA	33.3

Table 5

Statistics from Figure 7

A *SNR* calculation for the *entire* image based on eq. (40) is not directly available. The variations in the image brightnesses that lead to the large value of s ($=49.5$) are not, in general, due to noise but to the variation in local information. With the help of the region there is a way to estimate the *SNR*. We can use the s_R ($=4.0$) and the dynamic range, $a_{max} - a_{min}$, for the image ($=241 - 56$) to calculate a global *SNR* ($=33.3$ dB). The underlying assumptions are that 1) the signal is approximately constant in that region and the variation in the region is therefore due to noise, and, 2) that the noise is the same over the entire image with a standard deviation given by $s_n = s_R$.

CONTOUR REPRESENTATIONS

When dealing with a region or object, several compact representations are available that can facilitate manipulation of and measurements on the object. In each case we assume that we begin with an image representation of the object as shown in Figure 8a,b. Several techniques exist to represent the region or object by describing its contour.

Chain code

This representation is based upon the work of Freeman [11]. We follow the contour in a clockwise manner and keep track of the directions as we go from one

contour pixel to the next. For the standard implementation of the chain code we consider a contour pixel to be an object pixel that has a background (non-object) pixel as one or more of its 4-connected neighbors. See Figures 3a and 8c.

The codes associated with eight possible directions are the chain codes and, with x as the current contour pixel position, the codes are generally defined as:

$$\text{Chain codes} = \begin{matrix} & 3 & 2 & 1 \\ & 4 & x & 0 \\ & 5 & 6 & 7 \end{matrix} \quad (43)$$

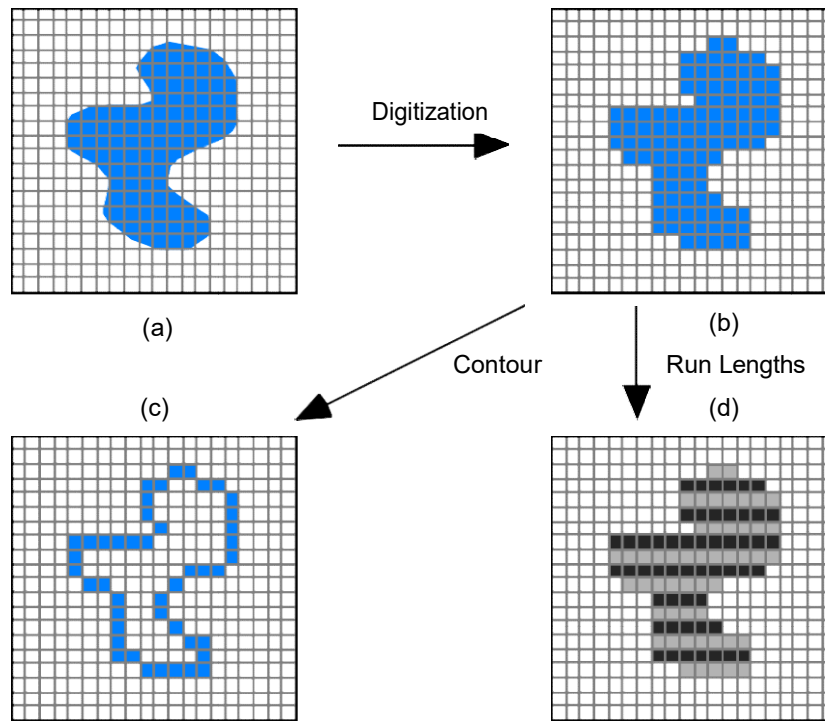


Figure 8: Region (shaded) as it is transformed from (a) continuous to (b) discrete form and then considered as a (c) contour or (d) run lengths illustrated in alternating colors.

Chain code properties

- Even codes $\{0,2,4,6\}$ correspond to horizontal and vertical directions; odd codes $\{1,3,5,7\}$ correspond to the diagonal directions.
- Each code can be considered as the angular direction, in multiples of 45° , that we must move to go from one contour pixel to the next.
- The absolute coordinates $[m,n]$ of the first contour pixel (e.g. top, leftmost) together with the chain code of the contour represent a complete description of the discrete region contour.

- When there is a change between two consecutive chain codes, then the contour has changed direction. This point is defined as a *corner*.

“Crack” code

An alternative to the chain code for contour encoding is to use neither the contour pixels associated with the object nor the contour pixels associated with background but rather the line, the “crack”, in between. This is illustrated with an enlargement of a portion of Figure 8 in Figure 9.

The “crack” code can be viewed as a chain code with four possible directions instead of eight.

1

$$\text{Crack codes} = \begin{matrix} & 1 \\ 2 & x & 0 \\ & 3 \end{matrix}$$

(44)

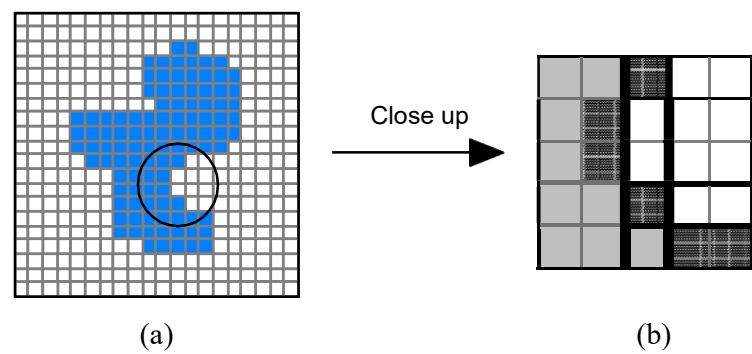


Figure 9: (a) Object including part to be studied. (b) Contour pixels as used in the chain code are diagonally shaded. The “crack” is shown with the thick black line.

The chain code for the enlarged section of Figure 9b, from top to bottom, is {5,6,7,7,0}. The crack code is {3,2,3,3,0,3,0,0}.

Run codes

A third representation is based on coding the consecutive pixels along a row—a run—that belong to an object by giving the starting position of the run and the ending position of the run. Such runs are illustrated in Figure 8d. There are a number of alternatives for the precise definition of the positions. Which alternative should be used depends upon the application and thus will not be discussed here.

4. Perception

Many image processing applications are intended to produce images that are to be viewed by human observers (as opposed to, say, automated industrial inspection.) It is therefore important to understand the characteristics and limitations of the human visual system—to understand the “receiver” of the 2D signals. At the outset it is important to realize that 1) the human visual system is not well understood, 2) no objective measure exists for judging the quality of an image that corresponds to human assessment of image quality, and, 3) the “typical” human observer does not exist. Nevertheless, research in perceptual psychology has provided some important insights into the visual system. See, for example, Stockham [12].

BRIGHTNESS SENSITIVITY

There are several ways to describe the sensitivity of the human visual system. To begin, let us assume that a homogeneous region in an image has an intensity as a function of wavelength (color) given by $I(\lambda)$. Further let us assume that $I(\lambda) = I_o$, a constant.

Wavelength sensitivity

The perceived intensity as a function of λ , the spectral sensitivity, for the “typical observer” is shown in Figure 10 [13].

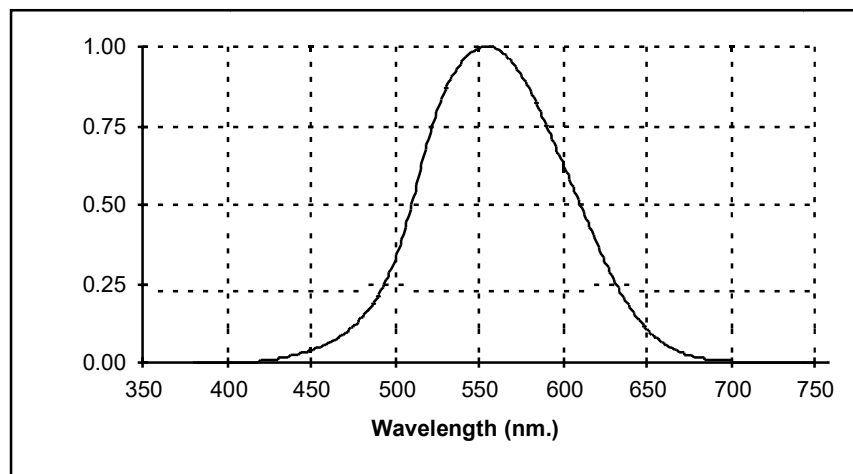


Figure 10: Spectral Sensitivity of the “typical” human observer

Stimulus sensitivity

If the constant intensity (brightness) I_o is allowed to vary then, to a good approximation, the visual response, R , is proportional to the logarithm of the intensity. This is known as the Weber–Fechner law:

$$R = \log(I_o) \quad (45)$$

The implications of this are easy to illustrate. Equal *perceived* steps in brightness, $\otimes R = k$, require that the physical brightness (the stimulus) increases exponentially. This is illustrated in Figure 11ab.

A horizontal line through the top portion of Figure 11a shows a linear increase in objective brightness (Figure 11b) but a logarithmic increase in subjective brightness. A horizontal line through the bottom portion of Figure 11a shows an exponential increase in objective brightness (Figure 11b) but a linear increase in subjective brightness.

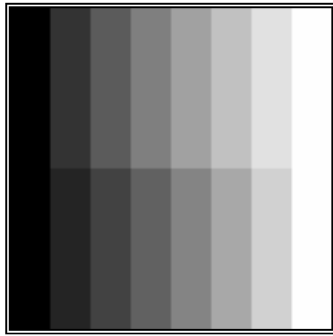


Figure 11a

(top) Brightness step $\otimes I = k$

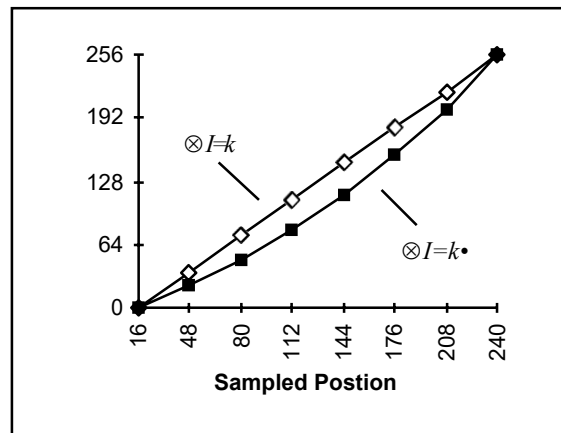


Figure 11b

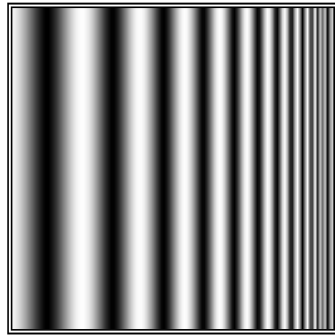
Actual brightnesses plus interpolated values

(bottom) Brightness step $\otimes I = k \cdot I$

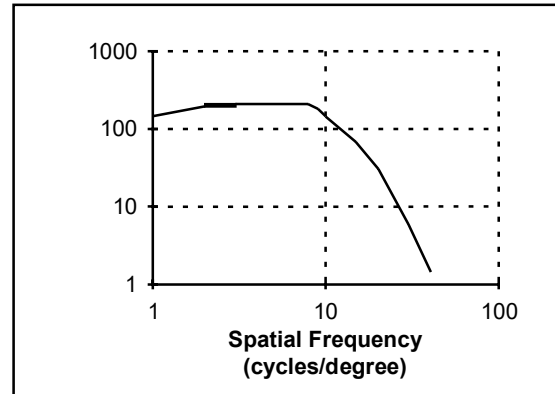
The *Mach band effect* is visible in Figure 11a. Although the physical brightness is constant across each vertical stripe, the human observer perceives an “undershoot” and “overshoot” in brightness at what is physically a step edge. Thus, just before the step, we see a slight decrease in brightness compared to the true physical value. After the step we see a slight overshoot in brightness compared to the true physical value. The total effect is one of increased, local, *perceived* contrast at a step edge in brightness.

SPATIAL FREQUENCY SENSITIVITY

If the constant intensity (brightness) I_o is replaced by a sinusoidal grating with increasing spatial frequency (Figure 12a), it is possible to determine the spatial frequency sensitivity. The result is shown in Figure 12b [14, 15].

**Figure 12a**

Sinusoidal test grating

**Figure 12b**

Spatial frequency sensitivity

To translate these data into common terms, consider an “ideal” computer monitor at a viewing distance of 50 cm. The spatial frequency that will give maximum response is at 10 cycles per degree. (See Figure 12b.) The one degree at 50 cm translates to $50 \tan(1^\circ) = 0.87$ cm on the computer screen. Thus the spatial frequency of maximum response $f_{\max} = 10 \text{ cycles}/0.87 \text{ cm} = 11.46 \text{ cycles/cm}$ at this viewing distance. Translating this into a general formula gives:

$$f_{\max} = \frac{10}{d \cdot \tan(1^\circ)} = \frac{572.9}{d} \text{ cycles/cm} \quad (46)$$

where d = viewing distance measured in cm.

COLOR SENSITIVITY

Human color perception is an exceedingly complex topic. As such we can only present a brief introduction here. The physical perception of color is based upon three color pigments in the retina.

Standard observer

Based upon psychophysical measurements, standard curves have been adopted by the CIE (Commission Internationale de l’Eclairage) as the sensitivity curves for the “typical” observer for the three “pigments” $\bar{x}(\lambda)$, $\bar{y}(\lambda)$, and $\bar{z}(\lambda)$. These are shown in Figure 13. These are not the *actual* pigment absorption characteristics found in the “standard” human retina but rather sensitivity curves derived from actual data [10].

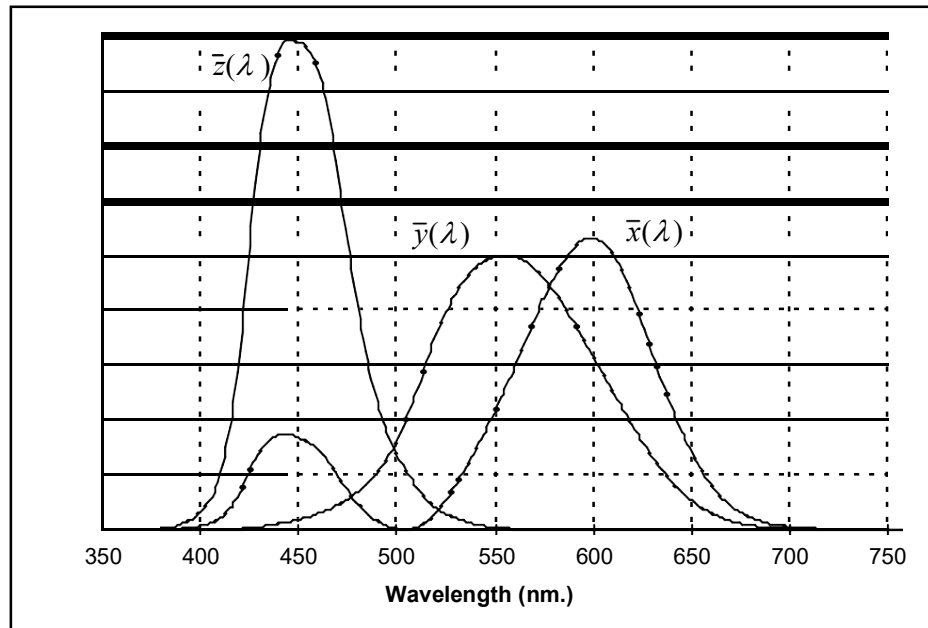


Figure 13: Standard observer spectral sensitivity curves.

For an arbitrary homogeneous region in an image that has an intensity as a function of wavelength (color) given by $I(\lambda)$, the three responses are called the *tristimulus values*:

$$X = \int_0^{\infty} I(\lambda) \bar{x}(\lambda) d\lambda \quad Y = \int_0^{\infty} I(\lambda) \bar{y}(\lambda) d\lambda \quad Z = \int_0^{\infty} I(\lambda) \bar{z}(\lambda) d\lambda \quad (47)$$

CIE chromaticity coordinates

The *chromaticity coordinates* which describe the perceived color information are defined as:

$$x = \frac{X}{X + Y + Z} \quad y = \frac{Y}{X + Y + Z} \quad z = 1 - (x + y) \quad (48)$$

The red chromaticity coordinate is given by x and the green chromaticity coordinate by y . The tristimulus values are linear in $I(\lambda)$ and thus the absolute intensity information has been lost in the calculation of the chromaticity coordinates $\{x, y\}$. All color distributions, $I(\lambda)$, that appear to an observer as having the same color will have the same chromaticity coordinates.

If we use a tunable source of pure color (such as a dye laser), then the intensity can be modeled as $I(\lambda) = \delta(\lambda - \lambda_0)$ with $\delta(\bullet)$ as the impulse function. The collection of chromaticity coordinates $\{x, y\}$ that will be generated by varying λ_0 gives the *CIE chromaticity triangle* as shown in Figure 14.

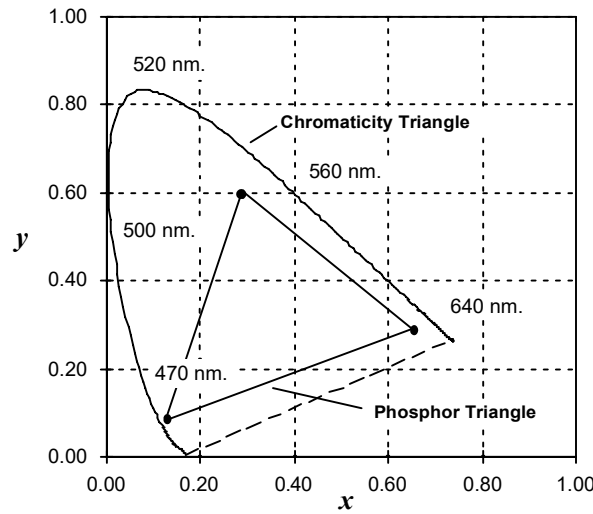


Figure 14: Chromaticity diagram containing the *CIE chromaticity triangle* associated with pure spectral colors and the triangle associated with CRT phosphors.

Pure spectral colors are along the boundary of the chromaticity triangle. All other colors are inside the triangle. The chromaticity coordinates for some standard sources are given in Table 6.

Source	x	y
Fluorescent lamp @ 4800 °K	0.35	0.37
Sun @ 6000 °K	0.32	0.33
Red Phosphor (europium yttrium vanadate)	0.68	0.32
Green Phosphor (zinc cadmium sulfide)	0.28	0.60
Blue Phosphor (zinc sulfide)	0.15	0.07

Table 6: Chromaticity coordinates for standard sources.

The description of color on the basis of chromaticity coordinates not only permits an analysis of color but provides a synthesis technique as well. Using a mixture of two color sources, it is possible to generate any of the colors along the line connecting their respective chromaticity coordinates. Since we cannot have a negative number of photons, this means the mixing coefficients must be positive. Using three color sources such as the red, green, and blue phosphors on CRT monitors leads to the set of colors defined by the *interior* of the “phosphor triangle” shown in Figure 14.

The formulas for converting from the tristimulus values (X, Y, Z) to the well-known CRT colors (R, G, B) and back are given by:

$$\begin{pmatrix} R \\ G \\ B \end{pmatrix} = \begin{pmatrix} 1.9107 & -0.5326 & -0.2883 \\ -0.9843 & 1.9984 & -0.0283 \\ 0.0583 & -0.1185 & 0.8986 \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \end{pmatrix} \quad (49)$$

and

$$\begin{pmatrix} X \\ Y \\ Z \end{pmatrix} = \begin{pmatrix} 0.6067 & 0.1736 & 0.2001 \\ 0.2988 & 0.5868 & 0.1143 \\ 0.0000 & 0.0661 & 1.1149 \end{pmatrix} \begin{pmatrix} R \\ G \\ B \end{pmatrix} \quad (50)$$

As long as the position of a desired color (X, Y, Z) is inside the phosphor triangle in Figure 14, the values of R , G , and B as computed by eq. (49) will be positive and can therefore be used to drive a CRT monitor.

It is incorrect to assume that a small displacement anywhere in the chromaticity diagram (Figure 14) will produce a proportionally small change in the *perceived* color. An empirically-derived chromaticity space where this property is approximated is the (u', v') space:

$$\begin{aligned} u' &= \frac{4x}{-2x + 12y + 3} & v' &= \frac{9y}{-2x + 12y + 3} \\ \text{and} & & & \\ x &= \frac{9u'}{6u' - 16v' + 12} & y &= \frac{4v'}{6u' - 16v' + 12} \end{aligned} \quad (51)$$

Small changes almost anywhere in the (u', v') chromaticity space produce equally small changes in the perceived colors.

OPTICAL ILLUSIONS

The description of the human visual system presented above is couched in standard engineering terms. This could lead one to conclude that there is sufficient knowledge of the human visual system to permit modeling the visual system with standard system analysis techniques. Two simple examples of optical illusions, shown in Figure 15, illustrate that this system approach would be a gross oversimplification. Such models should only be used with extreme care.

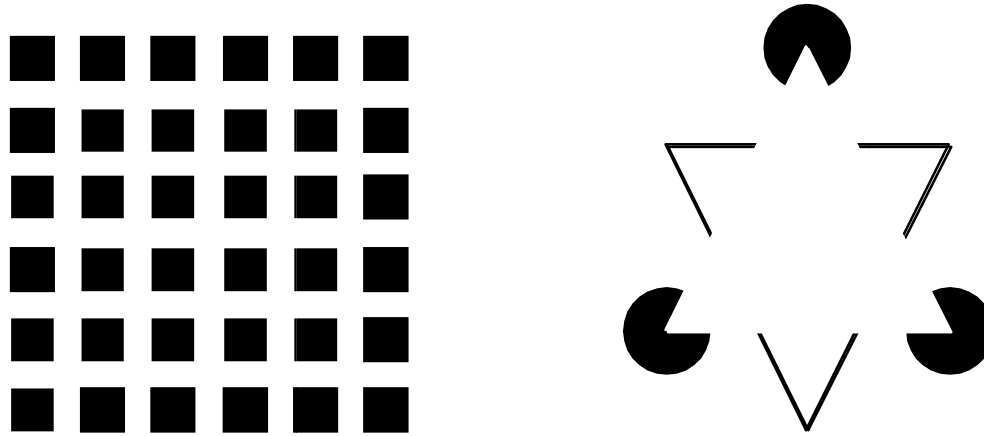


Figure 15: Optical Illusions

The left illusion induces the illusion of gray values in the eye that the brain “knows” does not exist. Further, there is a sense of dynamic change in the image due, in part, to the saccadic movements of the eye. The right illusion, Kanizsa’s triangle, shows enhanced contrast and false contours [14] neither of which can be explained by the system-oriented aspects of visual perception described above.

5. Image Sampling

Converting from a continuous image $a(x,y)$ to its digital representation $b[m,n]$ requires the process of sampling. In the ideal sampling system $a(x,y)$ is multiplied by an ideal 2D impulse train:

$$\begin{aligned} b_{ideal}[m,n] &= a(x,y) \cdot \sum_{m=-\infty}^{+\infty} \sum_{n=-\infty}^{+\infty} \delta(x - mX_o, y - nY_o) \\ &= \sum_{m=-\infty}^{+\infty} \sum_{n=-\infty}^{+\infty} a(mX_o, nY_o) \delta(x - mX_o, y - nY_o) \end{aligned} \quad (52)$$

where X_o and Y_o are the sampling distances or intervals and $\delta(\bullet, \bullet)$ is the ideal impulse function. (At some point, of course, the impulse function $\delta(x,y)$ is converted to the discrete impulse function $\delta[m,n]$.) *Square sampling* implies that $X_o = Y_o$. Sampling with an impulse function corresponds to sampling with an infinitesimally small point. This, however, does not correspond to the usual situation as illustrated in Figure 1. To take the effects of a *finite* sampling aperture $p(x,y)$ into account, we can modify the sampling model as follows:

$$b[m,n] = (a(x,y) \otimes p(x,y)) \cdot \sum_{m=-\infty}^{+\infty} \sum_{n=-\infty}^{+\infty} \delta(x - mX_o, y - nY_o) \quad (53)$$

The combined effect of the aperture and sampling are best understood by examining the Fourier domain representation.

$$B(\wedge, \Psi) = \frac{1}{4\pi^2} \sum_{m=-\infty}^{+\infty} \sum_{n=-\infty}^{+\infty} A(\wedge - m\wedge_s, \Psi - n\Psi_s) \cdot P(\wedge - m\wedge_s, \Psi - n\Psi_s) \quad (54)$$

where $\wedge_s = 2\pi/X_o$ is the sampling frequency in the x direction and $\Psi_s = 2\pi/Y_o$ is the sampling frequency in the y direction. The aperture $p(x,y)$ is frequently square, circular, or Gaussian with the associated $P(\wedge, \Psi)$. (See Table 4.) The periodic nature of the spectrum, described in eq. (21) is clear from eq. (54).

SAMPLING DENSITY FOR IMAGE PROCESSING

To prevent the possible *aliasing* (overlapping) of spectral terms that is inherent in eq. (54) two conditions must hold:

- *Bandlimited* $A(u,v)$ –

$$|A(u, v)| \equiv 0 \quad \text{for} \quad |u| > u_c \quad \text{and} \quad |v| > v_c \quad (55)$$

- *Nyquist sampling frequency* –

$$\wedge_s > 2 \cdot u_c \quad \text{and} \quad \Psi_s > 2 \cdot v_c \quad (56)$$

where u_c and v_c are the *cutoff frequencies* in the x and y direction, respectively. Images that are acquired through lenses that are circularly-symmetric, aberration-free, and diffraction-limited will, in general, be bandlimited. The lens acts as a lowpass filter with a cutoff frequency in the frequency domain (eq. (11)) given by:

$$u_c = v_c = \frac{2NA}{\lambda} \quad (57)$$

where NA is the numerical aperture of the lens and λ is the shortest wavelength of light used with the lens [16]. If the lens does not meet one or more of these assumptions then it will still be bandlimited but at lower cutoff frequencies than those given in eq. (57). When working with the F-number (F) of the optics instead of the NA and in air (with *index of refraction* = 1.0), eq. (57) becomes:

$$u_c = v_c = \frac{2}{\lambda} \sqrt{4F^2 + 1} \quad (58)$$

5.1.1 Sampling aperture

The aperture $p(x,y)$ described above will have only a marginal effect on the final signal if the two conditions eqs. (56) and (57) are satisfied. Given, for example, the distance between samples X_o equals Y_o and a sampling aperture that is not wider than X_o , the effect on the overall spectrum—due to the $A(u,v)P(u,v)$ behavior implied by eq.(53)—is illustrated in Figure 16 for square and Gaussian apertures.

The spectra are evaluated along one axis of the 2D Fourier transform. The Gaussian aperture in Figure 16 has a width such that the sampling interval X_o contains $\pm 3\sigma$ (99.7%) of the Gaussian. The rectangular apertures have a width such that one occupies 95% of the sampling interval and the other occupies 50% of the sampling interval. The 95% width translates to a *fill factor* of 90% and the 50% width to a *fill factor* of 25%. The *fill factor* is discussed in Section 7.5.2.

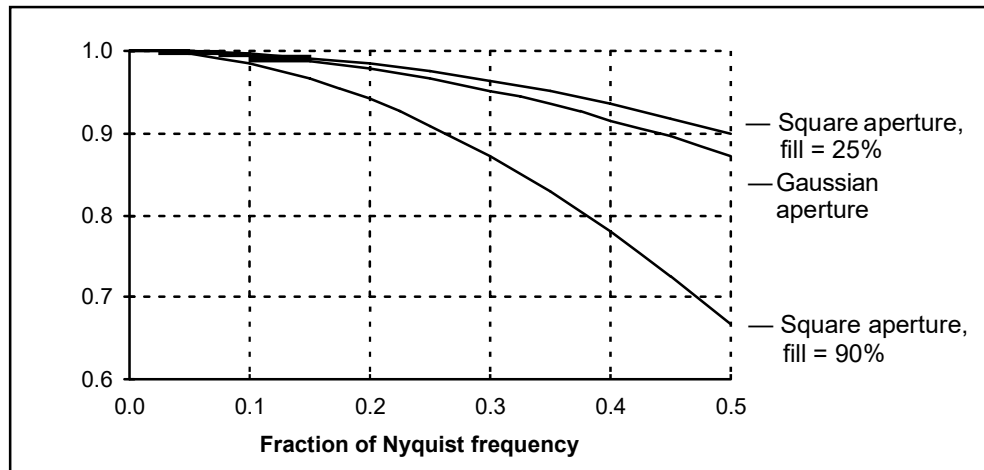


Figure 16: Aperture spectra $P(u,v=0)$ for frequencies up to half the Nyquist frequency. For explanation of “fill” see text.

SAMPLING DENSITY FOR IMAGE ANALYSIS

The “rules” for choosing the sampling density when the goal is image analysis—as opposed to image processing—are different. The fundamental difference is that the digitization of objects in an image into a collection of pixels introduces a form of spatial quantization noise that is not bandlimited. This leads to the following results for the choice of sampling density when one is interested in the measurement of area and (perimeter) length.

Sampling for area measurements

Assuming square sampling, $X_o = Y_o$ and the unbiased algorithm for estimating area which involves simple pixel counting, the CV (see eq. (38)) of the area measurement is related to the sampling density by [17]:

$$2D: \lim_{S \rightarrow \infty} CV(S) = k_2 S^{-3/2} \quad 3D: \lim_{S \rightarrow \infty} CV(S) = k_3 S^{-2} \quad (59)$$

and in D dimensions:

$$\lim_{S \rightarrow \infty} CV(S) = k_D S^{-(D+1)/2} \quad (60)$$

where S is the number of samples *per object diameter*. In 2D the measurement is area, in 3D volume, and in D -dimensions hypervolume.

Sampling for length measurements

Again assuming square sampling and algorithms for estimating length based upon the Freeman chain-code representation (see Section 3.6.1), the CV of the length measurement is related to the sampling density *per unit length* as shown in Figure 17 (see [18, 19].)

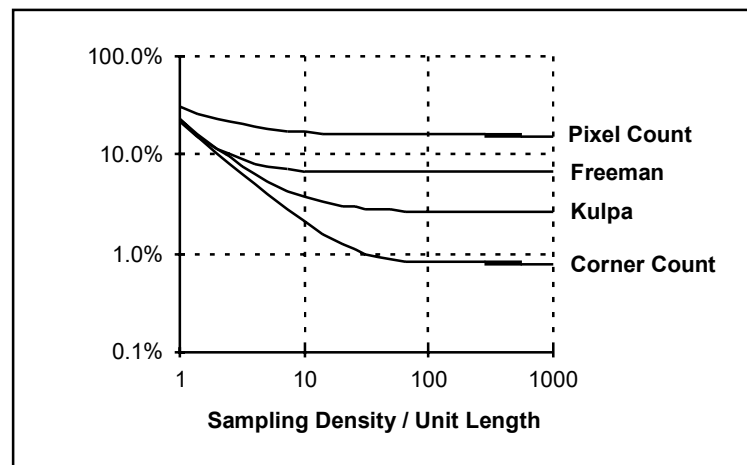


Figure 17: CV of length measurement for various algorithms.

The curves in Figure 17 were developed in the context of straight lines but similar results have been found for curves and closed contours. The specific formulas for length estimation use a chain code representation of a line and are based upon a linear combination of three numbers:

$$L = \alpha \cdot N_e + \beta \cdot N_o + \gamma \cdot N_c \quad (61)$$

where N_e is the number of even chain codes, N_o the number of odd chain codes, and N_c the number of corners. The specific formulas are given in Table 7.

<i>Coefficients Formula</i>	α	β	γ	<i>Reference</i>
Pixel count	1	1	0	[18]
Freeman	1	$\sqrt{2}$	0	[11]
Kulpa	0.9481	$0.9481 \cdot \sqrt{2}$	0	[20]
Corner count	0.980	1.406	-0.091	[21]

Table 7: Length estimation formulas based on chain code counts (N_e , N_o , N_c)

Conclusions on sampling

If one is interested in image processing, one should choose a sampling density based upon classical signal theory, that is, the Nyquist sampling theory. If one is interested in image analysis, one should choose a sampling density based upon the desired measurement accuracy (*bias*) and precision (*CV*). In a case of uncertainty, one should choose the higher of the two sampling densities (frequencies).

Contrast stretching

Frequently, an image is scanned in such a way that the resulting brightness values do not make full use of the available dynamic range. This can be easily observed in the histogram of the brightness values shown in Figure 6. By stretching the histogram over the available dynamic range we attempt to correct this situation. If the image is intended to go from brightness 0 to brightness 2^B-1 (see Section 2.1), then one generally maps the 0% value (or *minimum* as defined in Section 3.5.2) to the value 0 and the 100% value (or *maximum*) to the value 2^B-1 . The appropriate transformation is given by:

$$b[m, n] = (2^B - 1) \cdot \frac{a[m, n] - \text{minimum}}{\text{maximum} - \text{minimum}} \quad (77)$$

This formula, however, can be somewhat sensitive to outliers and a less sensitive and more general version is given by:

$$b[m, n] = \begin{cases} 0 & a[m, n] \leq p_{\text{low}} \% \\ (2^B - 1) \cdot \frac{a[m, n] - p_{\text{low}} \%}{p_{\text{high}} \% - p_{\text{low}} \%} & p_{\text{low}} \% < a[m, n] < p_{\text{high}} \% \\ (2^B - 1) & a[m, n] \geq p_{\text{high}} \% \end{cases} \quad (78)$$

In this second version one might choose the 1% and 99% values for $p_{\text{low}}\%$ and $p_{\text{high}}\%$, respectively, instead of the 0% and 100% values represented by eq. (77). It is also possible to apply the contrast-stretching operation on a regional basis using the histogram from a region to determine the appropriate limits for the algorithm. Note that in eqs. (77) and (78) it is possible to suppress the term 2^B-1 and simply normalize the brightness range to $0 \leq b[m,n] \leq 1$. This means representing the final pixel brightnesses as reals instead of integers but modern computer speeds and RAM capacities make this quite feasible.

Equalization

When one wishes to compare two or more images on a specific basis, such as texture, it is common to first normalize their histograms to a “standard” histogram. This can be especially useful when the images have been acquired under different circumstances. The most common histogram normalization technique is *histogram equalization* where one attempts to change the histogram through the use of a function $b = f(a)$ into a histogram that is constant for all brightness values. This would correspond to a brightness distribution where all values are equally probable. Unfortunately, for an arbitrary image, one can only approximate this result.

For a “suitable” function $f(\bullet)$ the relation between the input probability density function, the output probability density function, and the function $f(\bullet)$ is given by:

$$p_b(b)db = p_a(a)da \Rightarrow df = \frac{p_a(a)da}{p_b(b)} \quad (79)$$

From eq. (79) we see that “suitable” means that $f(\bullet)$ is differentiable and that $df/da \geq 0$. For histogram equalization we desire that $p_b(b) = \text{constant}$ and this means that:

$$f(a) = (2^B - 1) \cdot P(a) \quad (80)$$

where $P(a)$ is the probability *distribution* function defined in Section 3.5.1 and illustrated in Figure 6a. In other words, the *quantized* probability distribution function normalized from 0 to $2^B - 1$ is the look-up table required for histogram equalization. Figures 21a-c illustrate the effect of contrast stretching and histogram equalization on a standard image. The histogram equalization procedure can also be applied on a regional basis.



Figure 21a
Original



Figure 21b
Contrast Stretched



Figure 21c
Histogram Equalized

Other histogram-based operations

The histogram derived from a local region can also be used to drive local filters that are to be applied to that region. Examples include *minimum* filtering, *median* filtering, and *maximum* filtering [23]. The concepts minimum, median, and maximum were introduced in Figure 6. The filters based on these concepts will be presented formally in Sections 9.4.2 and 9.6.10.

MATHEMATICS-BASED OPERATIONS

We distinguish in this section between binary arithmetic and ordinary arithmetic. In the binary case there are two brightness values “0” and “1”. In the ordinary case we begin with 2^B brightness values or levels but the processing of the image can easily generate many more levels. For this reason many *software* systems provide 16 or 32 bit representations for pixel brightnesses in order to avoid problems with arithmetic overflow.

Binary operations

Operations based on binary (Boolean) arithmetic form the basis for a powerful set of tools that will be described here and extended in Section 9.6, mathematical morphology. The operations described below are point operations and thus admit a variety of efficient implementations including simple look-up tables. The standard notation for the basic set of binary operations is:

$$\begin{aligned}
 NOT & \quad c = \bar{a} \\
 OR & \quad c = a + b \\
 AND & \quad c = a \cdot b \\
 XOR & \quad c = a \oplus b = a \cdot \bar{b} + \bar{a} \cdot b \\
 SUB & \quad c = a \setminus b = a - b = a \cdot \bar{b}
 \end{aligned} \tag{81}$$

The implication is that each operation is applied on a pixel-by-pixel basis. For example, $c[m, n] = a[m, n] \cdot \bar{b}[m, n] \quad \forall m, n$. The definition of each operation is:

NOT	a	
0	1	
1	0	

↑
input

↑
output

OR	a	b	
	0	1	
0	0	1	
1	1	1	

AND	a	b	
	0	1	
0	0	0	
1	0	1	

XOR	a	b	
	0	1	
0	0	1	
1	1	0	

SUB	a	b	
	0	1	
0	0	0	
1	1	0	

(82)

These operations are illustrated in Figure 22 where the binary value “1” is shown in black and the value “0” in white.

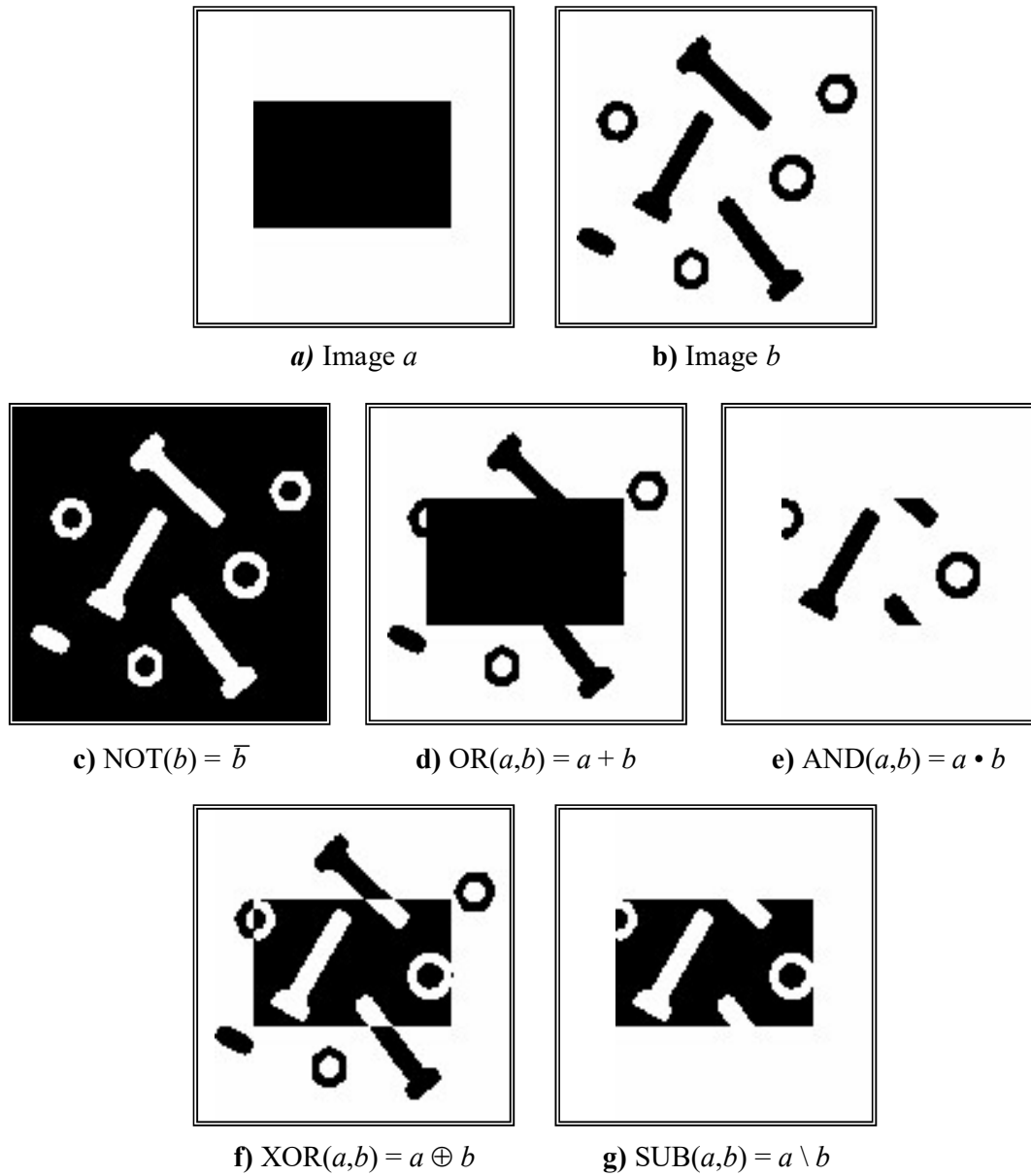


Figure 22: Examples of the various binary point operations.

The $\text{SUB}(\bullet)$ operation can be particularly useful when the image a represents a region-of-interest that we want to analyze systematically and the image b represents objects that, having been analyzed, can now be discarded, that is subtracted, from the region.

Arithmetic-based operations

The gray-value point operations that form the basis for image processing are based on ordinary mathematics and include:

<i>Operation</i>	<i>Definition</i>	<i>preferred data type</i>
ADD	$c = a + b$	integer
SUB	$c = a - b$	integer
MUL	$c = a \cdot b$	integer or floating point
DIV	$c = a / b$	floating point
LOG	$c = \log(a)$	floating point
EXP	$c = \exp(a)$	floating point
SQRT	$c = \sqrt{a}$	floating point
TRIG.	$c = \sin/\cos/\tan(a)$	floating point
INVERT	$c = (2^B - 1) - a$	integer

(83)

CONVOLUTION-BASED OPERATIONS

Convolution, the mathematical, *local* operation defined in Section 3.1 is central to modern image processing. The basic idea is that a window of some finite size and shape—the *support*—is scanned across the image. The output pixel value is the weighted sum of the input pixels within the window where the weights are the values of the filter assigned to every pixel of the window itself. The window with its weights is called the *convolution kernel*. This leads directly to the following variation on eq. (3). If the filter $h[j,k]$ is zero outside the (odd sized rectangular) window of size $J \times K$ centered around the origin $\{j=-J_0, -J_0+1, \dots, -1, 0, 1, \dots, J_0-1, J_0; k=-K_0, -K_0+1, \dots, -1, 0, 1, \dots, K_0-1, K_0\}$, then, using eq. (4), the convolution can be written as the following finite sum:

$$c[m,n] = a[m,n] \otimes h[m,n] = \sum_{j=-J_0}^{J_0} \sum_{k=-K_0}^{K_0} h[j,k] a[m-j, n-k] \quad (84)$$

This equation can be viewed as more than just a pragmatic mechanism for smoothing or sharpening an image. Further, while eq. (84) illustrates the local character of this operation, eqs. (10) and (24) suggest that the operation can be implemented through the use of the Fourier domain which requires a global operation, the Fourier transform. Both of these aspects will be discussed below.

Background

In a variety of image-forming systems an appropriate model for the transformation of the physical signal $a(x,y)$ into an electronic signal $c(x,y)$ is the convolution of the input signal with the impulse response of the sensor system. This system might consist of both an optical as well as an electrical sub-system. If

each of these systems can be treated as a linear, shift-invariant (*LSI*) system then the convolution model is appropriate. The definitions of these two, possible, system properties are given below:

$$\begin{array}{ll} \text{Linearity} - & \begin{array}{l} \text{If } a_1 \rightarrow c_1 \quad \text{and} \quad a_2 \rightarrow c_2 \\ \text{Then } w_1 \bullet a_1 + w_2 \bullet a_2 \rightarrow w_1 \bullet c_1 + w_2 \bullet c_2 \end{array} \end{array} \quad (85)$$

$$\begin{array}{ll} \text{Shift-Invariance} - & \begin{array}{l} \text{If } a(x, y) \rightarrow c(x, y) \\ \text{Then } a(x - x_o, y - y_o) \rightarrow c(x - x_o, y - y_o) \end{array} \end{array} \quad (86)$$

where w_1 and w_2 are arbitrary complex constants and x_o and y_o are coordinates corresponding to arbitrary spatial translations.

Two remarks are appropriate at this point. First, linearity implies (by choosing $w_1 = w_2 = 0$) that “zero in” gives “zero out”. The offset described in eq. (70) means that such camera signals are not the output of a linear system and thus (strictly speaking) the convolution result is not applicable. Fortunately, it is straightforward to correct for this non-linear effect. (See Section 10.1.)

Second, optical lenses with a magnification, M , other than $1\times$ are not shift invariant; a translation of 1 unit in the input image $a(x, y)$ produces a translation of M units in the output image $c(x, y)$. Due to the Fourier property described in eq. (25) this case can still be handled by linear system theory.

If an impulse point of light $\delta(x, y)$ is imaged through an LSI system then the impulse response of that system is called the *point spread function (PSF)*. The output image then becomes the convolution of the input image with the *PSF*. The Fourier transform of the *PSF* is called the *optical transfer function (OTF)*. For optical systems that are circularly-symmetric, aberration-free, and diffraction-limited the *PSF* is given by the Airy disk shown in Table 4–T.5. The *OTF* of the Airy disk is also presented in Table 4–T.5.

If the convolution window is not the diffraction-limited PSF of the lens but rather the effect of defocusing a lens then an appropriate model for $h(x, y)$ is a pill box of radius a as described in Table 4–T.3. The effect on a test pattern is illustrated in Figure 23.

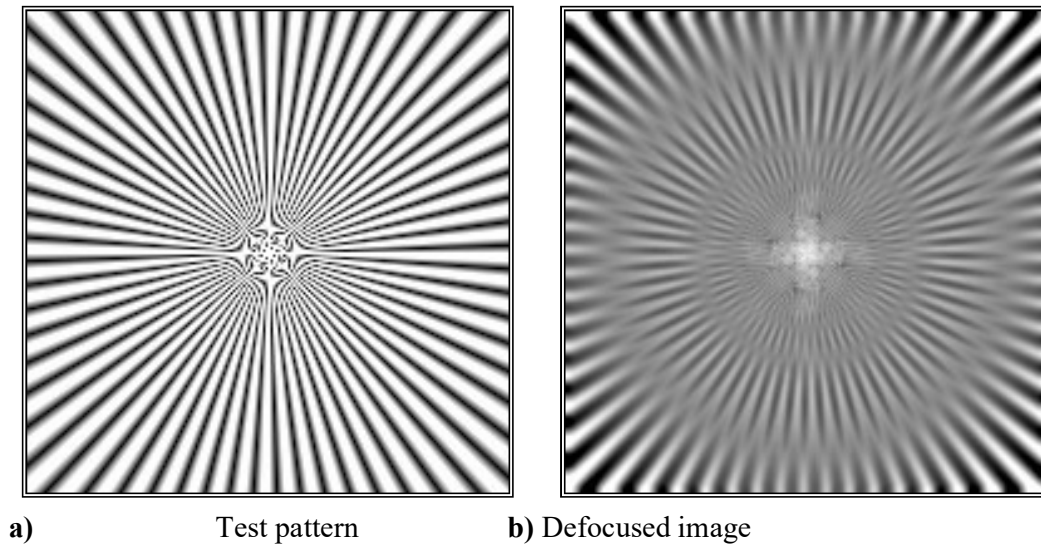


Figure 23: Convolution of test pattern with a pill box of radius $a=4.5$ pixels.

The effect of the defocusing is more than just simple blurring or smoothing. The almost periodic negative lobes in the transfer function in Table 4–T.3 produce a 180° phase shift in which black turns to white and vice-versa. The phase shift is clearly visible in Figure 23b.

Convolution in the spatial domain

In describing filters based on convolution we will use the following convention. Given a filter $h[j,k]$ of dimensions $J \times K = 2J_0+1 \times 2K_0+1$, we will consider the coordinate $[j=0,k=0]$ to be in the center of the filter matrix, \mathbf{h} . This is illustrated in Figure 24. The “center” is well-defined when the filter sizes are odd.

$$\mathbf{h} = \begin{matrix} & h[-J_0, -K_0] & \dots & \dots & h[0, -K_0] & \dots & \dots & h[J, -K_0]_{\infty} \\ & , & \% & \# & \# & \# & \% & \\ & , & & & & & & \infty \\ & , & \# & \dots & h[-1, -1] & h[0, -1] & h[1, -1] & \dots & \# & \infty \\ & , & h[-J_0, 0] & \dots & h[-1, 0] & h[0, 0] & h[1, 0] & \dots & h[J_0, 0] & \infty \\ & , & \# & \dots & h[-1, 1] & h[0, 1] & h[1, +1] & \dots & \# & \infty \\ & , & \# & \dots & h[-1, 1] & h[0, 1] & h[1, +1] & \dots & \# & \infty \\ & , & h[-J_0, K_0] & \% & \# & \# & \# & \% & h[J_0, K_0]_{\infty} \\ & \leq & \dots & \dots & h[0, K_0] & \dots & \dots & \dots & h[J_0, K_0]_{\infty} & f \end{matrix}$$

Figure 24: Coordinate system for describing $h[j,k]$

When we examine the convolution sum (eq. (84)) closely, several issues become evident.

- Evaluation of formula (84) for $m=n=0$ while rewriting the limits of the convolution sum based on the “centering” of $h[j,k]$ shows that values of $a[j,k]$ can be required that are outside the image boundaries:

$$c[0, 0] = \sum_{j=-J_0}^{+J_0} \sum_{k=-K_0}^{+K_0} h[j, k] a[-j, -k] \quad (87)$$

The question arises – what values should we assign to the image $a[m,n]$ for $m<0$, $m\geq M$, $n<0$, and $n\geq N$? There is no “answer” to this question. There are only alternatives among which we are free to choose assuming we understand the possible consequences of our choice. The standard alternatives are a) extend the images with a constant (possibly zero) brightness value, b) extend the image periodically, c) extend the image by mirroring it at its boundaries, or d) extend the values at the boundaries indefinitely. These alternatives are illustrated in Figure 25.

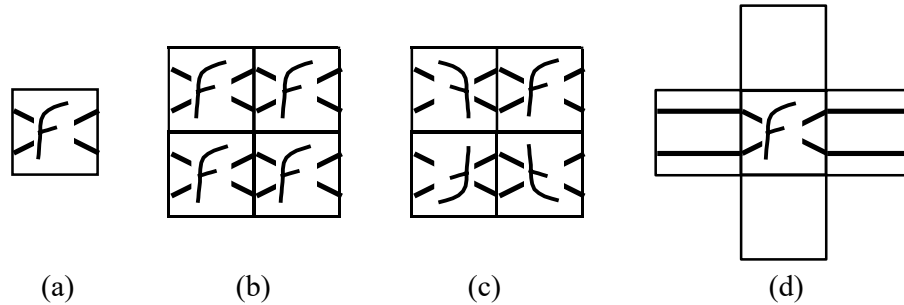


Figure 25: Examples of various alternatives to extend an image outside its formal boundaries. See text for explanation.

- When the convolution sum is written in the standard form (eq. (3)) for an image $a[m,n]$ of size $M \times N$:

$$c[m, n] = \sum_{j=0}^{M-1} \sum_{k=0}^{N-1} a[j, k] h[m-j, n-k] \quad (88)$$

we see that the convolution kernel $h[j,k]$ is mirrored around $j=k=0$ to produce $h[-j, -k]$ before it is translated by $[m,n]$ as indicated in eq. (88). While some convolution kernels in common use are symmetric in this respect, $h[j,k] = h[-j, -k]$, many are not. (See Section 9.5.) Care must therefore be taken in the implementation of filters with respect to the mirroring requirements.

- The computational complexity for a $J \times K$ convolution kernel implemented in the spatial domain on an image of $N \times N$ is $O(J \cdot K)$ where the complexity is measured *per pixel* on the basis of the number of multiplies-and-adds (MADDs).

- The value computed by a convolution that begins with integer brightnesses for $a[m,n]$ may produce a rational number or a floating point number in the result $c[m,n]$. Working exclusively with integer brightness values will, therefore, cause roundoff errors.

- Inspection of eq. (84) reveals another possibility for efficient implementation of convolution. If the convolution kernel $h[j,k]$ is *separable*, that is, if the kernel can be written as:

$$h[j, k] = h_{row}[k] \cdot h_{col}[j] \quad (89)$$

then the filtering can be performed as follows:

$$c[m, n] = \sum_{j=-J_0}^{J_0} \sum_{k=-K_0}^{K_0} h_{row}[k] a[m-j, n-k] h_{col}[j] \quad (90)$$

This means that instead of applying one, two-dimensional filter it is possible to apply two, one-dimensional filters, the first one in the k direction and the second one in the j direction. For an $N \times N$ image this, in general, reduces the computational complexity per pixel from $O(J \cdot K)$ to $O(J+K)$.

An alternative way of writing separability is to note that the convolution kernel (Figure 24) is a matrix \mathbf{h} and, if separable, \mathbf{h} can be written as:

$$\begin{aligned} [\mathbf{h}] &= [\mathbf{h}_{col}] \cdot [\mathbf{h}_{row}]^T \\ (J \times K) &= (J \times 1) \cdot (1 \times K) \end{aligned} \quad (91)$$

where “ T ” denotes the matrix transpose operation. In other words, \mathbf{h} can be expressed as the *outer product* of a column vector $[\mathbf{h}_{col}]$ and a row vector $[\mathbf{h}_{row}]$.

- For certain filters it is possible to find an *incremental implementation* for a convolution. As the convolution window moves over the image (see eq. (88)), the leftmost column of image data under the window is shifted out as a new column of image data is shifted in from the right. Efficient algorithms can take advantage of this [24] and, when combined with separable filters as described above, this can lead to algorithms where the computational complexity per pixel is $O(constant)$.

Convolution in the frequency domain

In Section 3.4 we indicated that there was an alternative method to implement the filtering of images through convolution. Based on eq. (24) it appears possible to achieve the same result as in eq. (84) by the following sequence of operations:

$$\begin{aligned}
& i) \quad \text{Compute } A(\wedge, \Psi) = \mathbf{F} \{a[m, n]\} \\
& ii) \quad \text{Multiply } A(\wedge, \Psi) \text{ by the precomputed } H(\wedge, \Psi) = \mathbf{F} \{h[m, n]\} \\
& iii) \quad \text{Compute the result } c[m, n] = \mathbf{F}^{-1} \{A(\wedge, \Psi) \cdot H(\wedge, \Psi)\}
\end{aligned} \tag{92}$$

• While it might seem that the “recipe” given above in eq. (92) circumvents the problems associated with direct convolution in the spatial domain—specifically, determining values for the image outside the boundaries of the image—the Fourier domain approach, in fact, simply “assumes” that the image is repeated periodically outside its boundaries as illustrated in Figure 25b. This phenomenon is referred to as *circular convolution*.

If circular convolution is not acceptable then the other possibilities illustrated in Figure 25 can be realized by embedding the image $a[m, n]$ and the filter $H(\wedge, \Psi)$ in larger matrices with the desired image extension mechanism for $a[m, n]$ being explicitly implemented.

• The computational complexity per pixel of the Fourier approach for an image of $N \times N$ and for a convolution kernel of $K \times K$ is $O(2 \log N)$ complex MADDs independent of K . Here we assume that $N > K$ and that N is a highly composite number such as a power of two. (See also 2.1.) This latter assumption permits use of the computationally-efficient Fast Fourier Transform (FFT) algorithm. Surprisingly then, the indirect route described by eq. (92) can be faster than the direct route given in eq. (84). This requires, in general, that $K^2 \gg 2 \log N$. The range of K and N for which this holds depends on the specifics of the implementation. For the machine on which this manuscript is being written and the specific image processing package that is being used, for an image of $N = 256$ the Fourier approach is faster than the convolution approach when $K \geq 15$. (It should be noted that in this comparison the direct convolution involves only integer arithmetic while the Fourier domain approach requires complex floating point arithmetic.)

SMOOTHING OPERATIONS

These algorithms are applied in order to reduce noise and/or to prepare images for further processing such as segmentation. We distinguish between linear and non-linear algorithms where the former are amenable to analysis in the Fourier domain and the latter are not. We also distinguish between implementations based on a rectangular support for the filter and implementations based on a circular support for the filter.

Linear Filters

Several filtering algorithms will be presented together with the most useful supports.

- *Uniform filter* – The output image is based on a local averaging of the input filter where all of the values within the filter support have the same weight. In the continuous spatial domain (x,y) the *PSF* and transfer function are given in Table 4–T.1 for the rectangular case and in Table 4–T.3 for the circular (pill box) case. For the discrete spatial domain $[m,n]$ the filter values are the samples of the continuous domain case. Examples for the rectangular case ($J=K=5$) and the circular case ($R=2.5$) are shown in Figure 26.

$$\begin{aligned}
 h_{rect}[j, k] &= \frac{1}{25} \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix} & h_{circ}[j, k] &= \frac{1}{21} \begin{bmatrix} 0 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 0 \end{bmatrix}
 \end{aligned}$$

(a) Rectangular filter ($J=K=5$) (b) Circular filter ($R=2.5$)

Figure 26: Uniform filters for image smoothing

Note that in both cases the filter is normalized so that $\sum h[j,k] = 1$. This is done so that if the input $a[m,n]$ is a constant then the output image $c[m,n]$ is the same constant. The justification can be found in the Fourier transform property described in eq. (26). As can be seen from Table 4, both of these filters have transfer functions that have negative lobes and can, therefore, lead to phase reversal as seen in Figure 23. The square implementation of the filter is separable and incremental; the circular implementation is incremental [24, 25].

- *Triangular filter* – The output image is based on a local averaging of the input filter where the values within the filter support have differing weights. In general, the filter can be seen as the convolution of two (identical) uniform filters either rectangular or circular and this has direct consequences for the computational complexity [24, 25]. (See Table 13.) In the continuous spatial domain the *PSF* and transfer function are given in Table 4–T.2 for the rectangular support case and in Table 4–T.4 for the circular (pill box) support case. As seen in Table 4 the transfer functions of these filters do not have negative lobes and thus do not exhibit phase reversal.

Examples for the rectangular support case ($J=K=5$) and the circular support case ($R=2.5$) are shown in Figure 27. The filter is again normalized so that $\sum h[j,k]=1$.

$$\begin{array}{cc}
 \begin{array}{ccccc}
 & \gamma_1 & 2 & 3 & 2 & 1 \\
 & \gamma_2 & 4 & 6 & 4 & 2 \\
 h_{rect}[j,k] = \frac{1}{81} & \gamma_3 & 6 & 9 & 6 & 3 \\
 & \gamma_2 & 4 & 6 & 4 & 2 \\
 & \gamma_1 & 2 & 3 & 2 & 1
 \end{array} &
 \begin{array}{ccccc}
 & \gamma_0 & 0 & 1 & 0 & 0 \\
 & \gamma_1 & 2 & 2 & 2 & 0 \\
 h_{circ}[j,k] = \frac{1}{25} & \gamma_2 & 4 & 6 & 4 & 2 \\
 & \gamma_3 & 6 & 9 & 6 & 3 \\
 & \gamma_2 & 4 & 6 & 4 & 2 \\
 & \gamma_1 & 2 & 3 & 2 & 1
 \end{array} \\
 \text{(a)} \quad \text{Pyramidal filter (J=K=5)} & \text{(b)} \quad \text{Cone filter (R=2.5)}
 \end{array}$$

Figure 27: Triangular filters for image smoothing

• *Gaussian filter* – The use of the Gaussian kernel for smoothing has become extremely popular. This has to do with certain properties of the Gaussian (e.g. the central limit theorem, minimum space-bandwidth product) as well as several application areas such as edge finding and scale space analysis. The *PSF* and transfer function for the continuous space Gaussian are given in Table 4–T6. The Gaussian filter is separable:

$$\begin{aligned}
 h(x, y) = g_{2D}(x, y) &= \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{x^2}{2\sigma^2}} \cdot \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{y^2}{2\sigma^2}} \\
 &= g_{1D}(x) \cdot g_{1D}(y)
 \end{aligned} \quad (93)$$

There are four distinct ways to implement the Gaussian:

– Convolution using a finite number of samples (N_o) of the Gaussian as the convolution kernel. It is common to choose $N_o = \gamma 3\sigma$ or $\gamma 5\sigma$.

$$g_{1D}[n] = \begin{cases} \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{n^2}{2\sigma^2}} & |n| \leq N_o \\ 0 & |n| > N_o \end{cases} \quad (94)$$

– Repetitive convolution using a uniform filter as the convolution kernel.

$$\begin{aligned}
 g_{1D}[n] &\approx u[n] \otimes u[n] \otimes u[n] \\
 u[n] &= \begin{cases} \frac{1}{(2N_o + 1)} & |n| \leq N_o \\ 0 & |n| > N_o \end{cases} \quad (95)
 \end{aligned}$$

The actual implementation (in each dimension) is usually of the form:

$$c[n] = ((a[n] \otimes u[n]) \otimes u[n]) \otimes u[n] \quad (96)$$

This implementation makes use of the approximation afforded by the central limit theorem. For a desired σ with eq. (96), we use $N_o = Y\sigma'$ although this severely restricts our choice of σ 's to integer values.

– Multiplication in the frequency domain. As the Fourier transform of a Gaussian is a Gaussian (see Table –T.6), this means that it is straightforward to prepare a filter $H(\wedge, \Psi) = G_{2D}(\wedge, \Psi)$ for use with eq. (92). To avoid truncation effects in the frequency domain due to the infinite extent of the Gaussian it is important to choose a σ that is sufficiently large. Choosing $\sigma > k/\pi$ where $k = 3$ or 4 will usually be sufficient.

– Use of a recursive filter implementation. A recursive filter has an infinite impulse response and thus an infinite support. The separable Gaussian filter can be implemented [26] by applying the following recipe in each dimension when $\sigma \geq 0.5$.

- i) Choose the σ based on the desired goal of the filtering;
- ii) Determine the parameter q based on eq. (98);
- iii) Use eq. (99) to determine the filter coefficients $\{b_0, b_1, b_2, b_3, B\}$; (97)
- iv) Apply the forward difference equation, eq. (100);
- v) Apply the backward difference equation, eq. (101);

The relation between the desired σ and q is given by:

$$q = \begin{cases} .98711\sigma - 0.96330 & \sigma \geq 2.5 \\ 3.97156 - 4.14554 \sqrt{1 - .26891\sigma} & 0.5 \leq \sigma \leq 2.5 \end{cases} \quad (98)$$

The filter coefficients $\{b_0, b_1, b_2, b_3, B\}$ are defined by:

$$\begin{aligned} b_0 &= 1.57825 + (2.44413q) + (1.4281q^2) + (0.422205q^3) \\ b_1 &= (2.44413q) + (2.85619q^2) + (1.26661q^3) \\ b_2 &= -(1.4281q^2) - (1.26661q^3) \\ b_3 &= 0.422205q^3 \\ B &= 1 - (b_1 + b_2 + b_3) / b_0 \end{aligned} \quad (99)$$

The one-dimensional *forward difference equation* takes an input row (or column) $a[n]$ and produces an intermediate output result $w[n]$ given by:

$$w[n] = Ba[n] + (b_1w[n-1] + b_2w[n-2] + b_3w[n-3]) / b_0 \quad (100)$$

The one-dimensional *backward difference equation* takes the intermediate result $w[n]$ and produces the output $c[n]$ given by:

$$c[n] = Bw[n] + (b_1c[n+1] + b_2c[n+2] + b_3c[n+3]) / b_0 \quad (101)$$

The forward equation is applied from $n = 0$ up to $n = N - 1$ while the backward equation is applied from $n = N - 1$ down to $n = 0$.

The relative performance of these various implementation of the Gaussian filter can be described as follows. Using the *root-square error* $\sqrt{\sum_{n=-\infty}^{+\infty} |g[n|\sigma] - h[n]|^2}$ between a true, infinite-extent Gaussian, $g[n|\sigma]$, and an approximated Gaussian, $h[n]$, as a measure of accuracy, the various algorithms described above give the results shown in Figure. 28a. The relative speed of the various algorithms in shown in Figure 28b.

The root-square error measure is extremely conservative and thus all filters, with the exception of “Uniform 3×” for large σ , are sufficiently accurate. The recursive implementation is the fastest independent of σ ; the other implementations can be significantly slower. The FFT implementation, for example, is 3.1 times slower for $N=256$. Further, the FFT requires that N be a highly composite number.

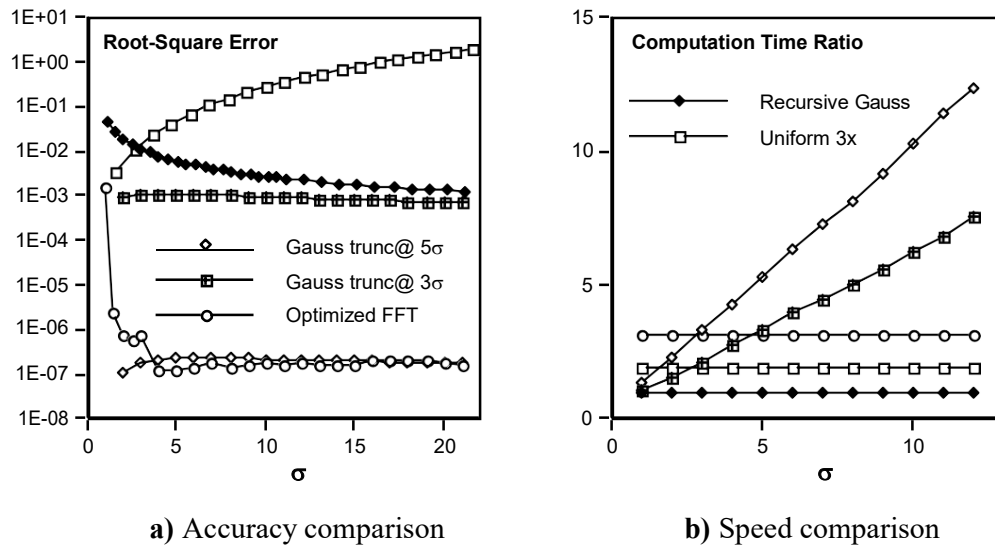


Figure 28: Comparison of various Gaussian algorithms with $N=256$.
The legend is spread across both graphs

• *Other* – The Fourier domain approach offers the opportunity to implement a variety of smoothing algorithms. The smoothing filters will then be *lowpass filters*. In general it is desirable to use a lowpass filter that has zero phase so as not to produce phase distortion when filtering the image. The importance of phase

was illustrated in Figures 5 and 23. When the frequency domain characteristics can be represented in an analytic form, then this can lead to relatively straightforward implementations of $H(\wedge, \Psi)$. Possible candidates include the lowpass filters “Airy” and “Exponential Decay” found in Table 4–T.5 and Table 4–T.8, respectively.

Non-Linear Filters

A variety of smoothing filters have been developed that are not linear. While they cannot, in general, be submitted to Fourier analysis, their properties and domains of application have been studied extensively.

- *Median filter* – The median statistic was described in Section 3.5.2. A median filter is based upon moving a window over an image (as in a convolution) and computing the output pixel as the median value of the brightnesses within the input window. If the window is $J \times K$ in size we can order the $J \cdot K$ pixels in brightness value from smallest to largest. If $J \cdot K$ is odd then the median will be the $(J \cdot K + 1)/2$ entry in the list of ordered brightnesses. Note that the value selected will be exactly equal to one of the existing brightnesses so that no roundoff error will be involved if we want to work exclusively with integer brightness values. The algorithm as it is described above has a generic complexity per pixel of $O(J \cdot K \cdot \log(J \cdot K))$. Fortunately, a fast algorithm (due to Huang et al. [23]) exists that reduces the complexity to $O(K)$ assuming $J \geq K$.

A useful variation on the theme of the median filter is the *percentile filter*. Here the center pixel in the window is replaced not by the 50% (median) brightness value but rather by the $p\%$ brightness value where $p\%$ ranges from 0% (the *minimum filter*) to 100% (the *maximum filter*). Values other than $(p=50)\%$ do not, in general, correspond to smoothing filters.

- *Kuwahara filter* – Edges play an important role in our perception of images (see Figure 15) as well as in the analysis of images. As such it is important to be able to smooth images without disturbing the sharpness and, if possible, the position of edges. A filter that accomplishes this goal is termed an *edge-preserving filter* and one particular example is the Kuwahara filter [27]. Although this filter can be implemented for a variety of different window shapes, the algorithm will be described for a square window of size $J = K = 4L + 1$ where L is an integer. The window is partitioned into four regions as shown in Figure 29.

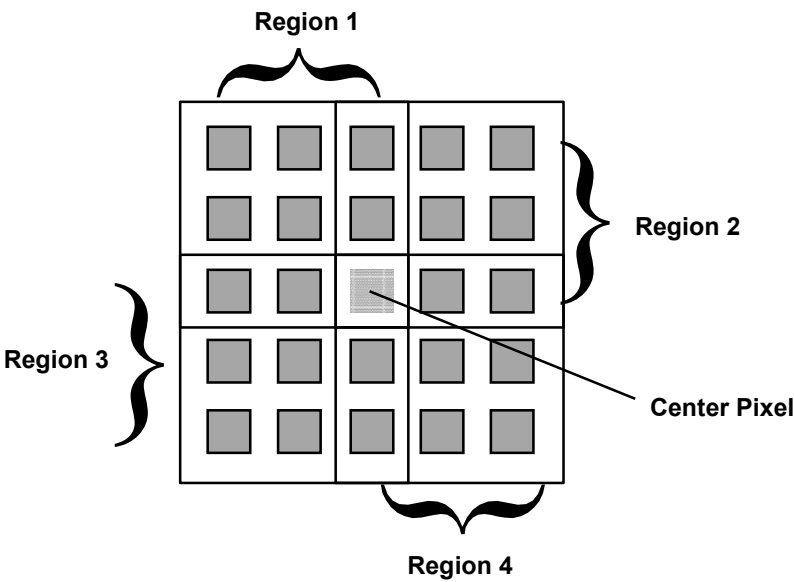


Figure 29: Four, square regions defined for the Kuwahara filter. In this example $L=1$ and thus $J=K=5$. Each region is $\lceil (J+1)/2 \rceil \times \lceil (K+1)/2 \rceil$.

In each of the four regions ($i=1,2,3,4$), the mean brightness, m_i in eq. (34), and the variance, s_i^2 in eq. (36), are measured. The output value of the center pixel in the window is the mean value of that region that has the smallest variance.

Summary of Smoothing Algorithms

The following table summarizes the various properties of the smoothing algorithms presented above. The filter size is assumed to be bounded by a rectangle of $J \times K$ where, without loss of generality, $J \geq K$. The image size is $N \times N$.

Algorithm	Domain	Type	Support	Separable / Incremental	Complexity/pixel
Uniform	Space	Linear	Square	Y / Y	$O(constant)$
Uniform	Space	Linear	Circular	N / Y	$O(K)$
Triangle	Space	Linear	Square	Y / N	$O(constant)^a$
Triangle	Space	Linear	Circular	N / N	$O(K)^a$
Gaussian	Space	Linear	∞^a	Y / N	$O(constant)^a$
Median	Space	Non-Linear	Square	N / Y	$O(K)^a$
Kuwahara	Space	Non-Linear	Square ^a	N / N	$O(J \cdot K)$
Other	Frequency	Linear	—	— / —	$O(\log N)$

Table 13: Characteristics of smoothing filters. ^aSee text for additional explanation.

Examples of the effect of various smoothing algorithms are shown in Figure 30.

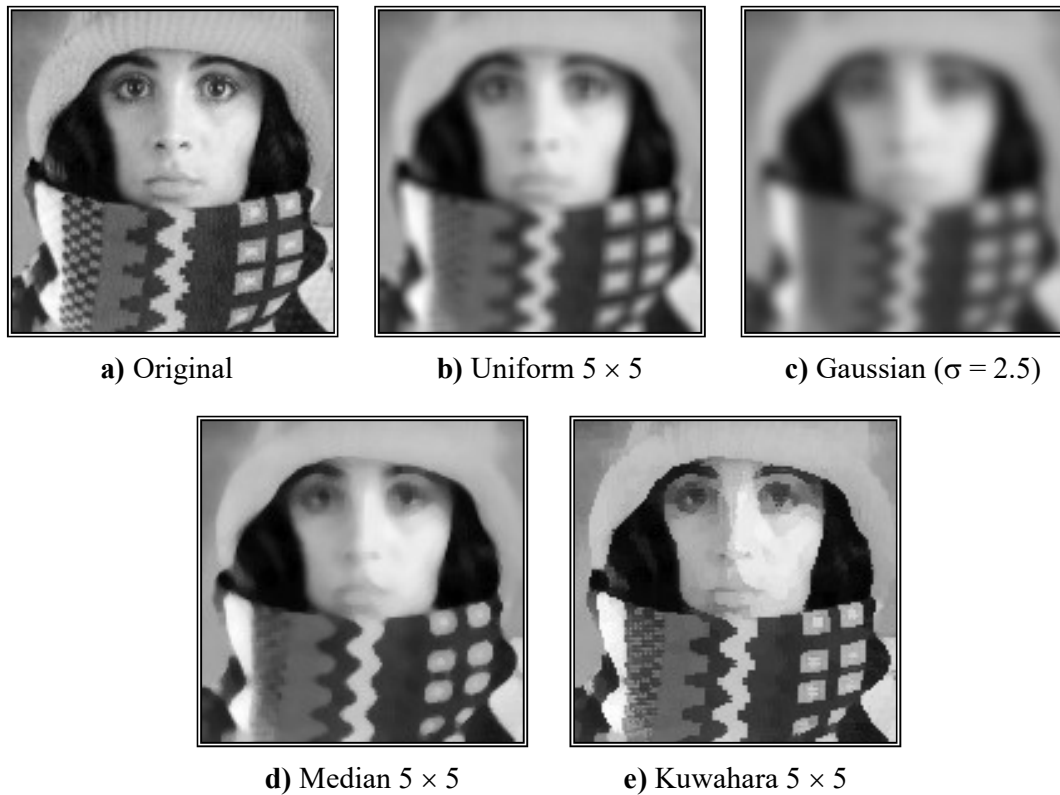


Figure 30: Illustration of various linear and non-linear smoothing filters

DERIVATIVE-BASED OPERATIONS

Just as smoothing is a fundamental operation in image processing so is the ability to take one or more spatial derivatives of the image. The fundamental problem is that, according to the mathematical definition of a derivative, this cannot be done. A digitized image is not a continuous function $a(x,y)$ of the spatial variables but rather a discrete function $a[m,n]$ of the integer spatial coordinates. As a result the algorithms we will present can only be seen as *approximations* to the true spatial derivatives of the original spatially-continuous image.

Further, as we can see from the Fourier property in eq. (27), taking a derivative multiplies the signal spectrum by either u or v . This means that high frequency noise will be emphasized in the resulting image. The general solution to this problem is to combine the derivative operation with one that suppresses high frequency noise, in short, smoothing in combination with the desired derivative operation.

First Derivatives

As an image is a function of two (or more) variables it is necessary to define the direction in which the derivative is taken. For the two-dimensional case we have

the horizontal direction, the vertical direction, or an arbitrary direction which can be considered as a combination of the two. If we use \mathbf{h}_x to denote a horizontal derivative filter (matrix), \mathbf{h}_y to denote a vertical derivative filter (matrix), and \mathbf{h}_θ to denote the arbitrary angle derivative filter (matrix), then:

$$[\mathbf{h}_\theta] = \cos \theta \cdot [\mathbf{h}_x] + \sin \theta \cdot [\mathbf{h}_y] \quad (102)$$

• *Gradient filters* – It is also possible to generate a vector derivative description as the *gradient*, $\nabla a[m,n]$, of an image:

$$\nabla a = \frac{\partial a}{\partial x} \mathbf{i}_x + \frac{\partial a}{\partial y} \mathbf{i}_y = \begin{pmatrix} h_x & a \end{pmatrix} \cdot \mathbf{i}_x + \begin{pmatrix} h_y & a \end{pmatrix} \cdot \mathbf{i}_y \quad (103)$$

where \mathbf{i}_x and \mathbf{i}_y are unit vectors in the horizontal and vertical direction, respectively. This leads to two descriptions:

$$\text{Gradient magnitude} - |\nabla a| = \sqrt{(h_x \otimes a)^2 + (h_y \otimes a)^2} \quad (104)$$

and

$$\text{Gradient direction} - \psi(\nabla a) = \arctan \frac{(h_y \otimes a)}{(h_x \otimes a)} \quad (105)$$

The gradient magnitude is sometimes approximated by:

$$\text{Approx. Gradient magnitude} - |\nabla a| \cong |h_x \otimes a| + |h_y \otimes a| \quad (106)$$

The final results of these calculations depend strongly on the choices of \mathbf{h}_x and \mathbf{h}_y . A number of possible choices for $(\mathbf{h}_x, \mathbf{h}_y)$ will now be described.

• *Basic derivative filters* – These filters are specified by:

$$\begin{aligned} i) \quad [\mathbf{h}_x] &= [\mathbf{h}_y]^t = \begin{bmatrix} 1 & -1 \end{bmatrix} \\ ii) \quad [\mathbf{h}_x] &= [\mathbf{h}_y]^t = \begin{bmatrix} 1 & 0 & -1 \end{bmatrix} \end{aligned} \quad (107)$$

where “ t ” denotes matrix transpose. These two filters differ significantly in their Fourier magnitude and Fourier phase characteristics. For the frequency range $0 \leq \omega \leq \pi$, these are given by:

$$\begin{aligned}
 i) \quad [\mathbf{h}] &= \begin{bmatrix} 1 & -1 \\ 1 & 0 \end{bmatrix} \leftrightarrow \begin{matrix} F \\ F \end{matrix} |H(\wedge)| = 2|\sin(\wedge/2)|; \varphi(\wedge) = (\pi - \wedge)/2 \\
 ii) \quad [\mathbf{h}] &= \begin{bmatrix} 1 & 0 & -1 \end{bmatrix} \leftrightarrow |H(\wedge)| = 2|\sin \wedge|; \quad \varphi(\wedge) = \pi/2
 \end{aligned} \quad (108)$$

The second form (ii) gives suppression of high frequency terms ($\wedge \approx \pi$) while the first form (i) does not. The first form leads to a phase shift; the second form does not.

• *Prewitt gradient filters* – These filters are specified by:

$$\begin{aligned}
 [\mathbf{h}_x] &= \frac{1}{3} \begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix} = \frac{1}{3} \begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix} \\
 [\mathbf{h}_{y/f}] &= \frac{1}{3} \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \end{bmatrix} = \frac{1}{3} \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \end{bmatrix}
 \end{aligned} \quad (109)$$

Both \mathbf{h}_x and \mathbf{h}_y are separable. Beyond the computational implications are the implications for the analysis of the filter. Each filter takes the derivative in one direction using eq. (107)ii and smoothes in the orthogonal direction using a one-dimensional version of a *uniform* filter as described in Section 9.4.1.

• *Sobel gradient filters* – These filters are specified by:

$$\begin{aligned}
 [\mathbf{h}_x] &= \frac{1}{4} \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \end{bmatrix} = \frac{1}{4} \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \end{bmatrix} \\
 [\mathbf{h}_{y/f}] &= \frac{1}{4} \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \end{bmatrix} = \frac{1}{4} \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \end{bmatrix}
 \end{aligned} \quad (110)$$

Again, \mathbf{h}_x and \mathbf{h}_y are separable. Each filter takes the derivative in one direction using eq. (107)ii and smoothes in the orthogonal direction using a one-dimensional version of a *triangular* filter as described in Section 9.4.1.

• *Alternative gradient filters* – The variety of techniques available from one-dimensional signal processing for the design of digital filters offers us powerful tools for designing one-dimensional versions of \mathbf{h}_x and \mathbf{h}_y .

McClellan filter design algorithm, for example, we can choose the frequency bands where we want the derivative to be taken and the frequency bands where we want the noise to be suppressed. The algorithm will then produce a real, odd filter with a minimum length that meets the specifications.

As an example, if we want a filter that has derivative characteristics in a passband (with weight 1.0) in the frequency range $0.0 \leq \omega \leq 0.3\pi$ and a stopband (with weight 3.0) in the range $0.32\pi \leq \omega \leq \pi$, then the algorithm produces the following optimized seven sample filter:

$$[\mathbf{h}_x] = \frac{1}{16348} \begin{bmatrix} -3571 & 8212 & -15580 & 0 & 15580 & -8212 & 3571 \end{bmatrix} \quad (111)$$

The gradient can then be calculated as in eq. (103).

• *Gaussian gradient filters* – In modern digital image processing one of the most common techniques is to use a Gaussian filter (see Section 9.4.1) to accomplish the required smoothing and one of the derivatives listed in eq. (107). Thus, we might first apply the recursive Gaussian in eq. (97) followed by eq. (107)ii to achieve the desired, smoothed derivative filters \mathbf{h}_x and \mathbf{h}_y . Further, for computational efficiency, we can combine these two steps as:

$$w[n] = \frac{B}{2} (a[n+1] - a[n-1]) + (b w[n-1] + b w[n-2] + b w[n-3]) / b \quad (112)$$

$$c[n] = Bw[n] + (b_1 c[n+1] + b_2 c[n+2] + b_3 c[n+3]) / b_0$$

where the various coefficients are defined in eq. (99). The first (forward) equation is applied from $n = 0$ up to $n = N - 1$ while the second (backward) equation is applied from $n = N - 1$ down to $n = 0$.

Within the class of linear filters, the optimal filter for restoration in the presence of noise is given by the *Wiener filter* [2]. The word “optimal” is used here in the sense of minimum mean-square error (*mse*). Because the square root operation is monotonic increasing, the optimal filter also minimizes the root mean-square error (*rms*). The Wiener filter is characterized in the Fourier domain and for additive noise that is independent of the signal it is given by:

$$H_W(u, v) = \frac{S_{aa}(u, v)}{S_{aa}(u, v) + S_{nn}(u, v)} \quad (194)$$

where $S_{aa}(u, v)$ is the power spectral density of an ensemble of random images $\{a[m, n]\}$ and $S_{nn}(u, v)$ is the power spectral density of the random noise. If we have a single image then $S_{aa}(u, v) = |A(u, v)|^2$. In practice it is unlikely that the power spectral density of the uncontaminated image will be available. Because many images have a similar power spectral density that can be modeled by Table 4–T.8, that model can be used as an estimate of $S_{aa}(u, v)$.

A comparison of the five different techniques described above is shown in Figure 49. The Wiener filter was constructed directly from eq. (113) because the image spectrum and the noise spectrum were known. The parameters for the other filters were determined choosing that value (either σ or window size) that led to the minimum *rms*.

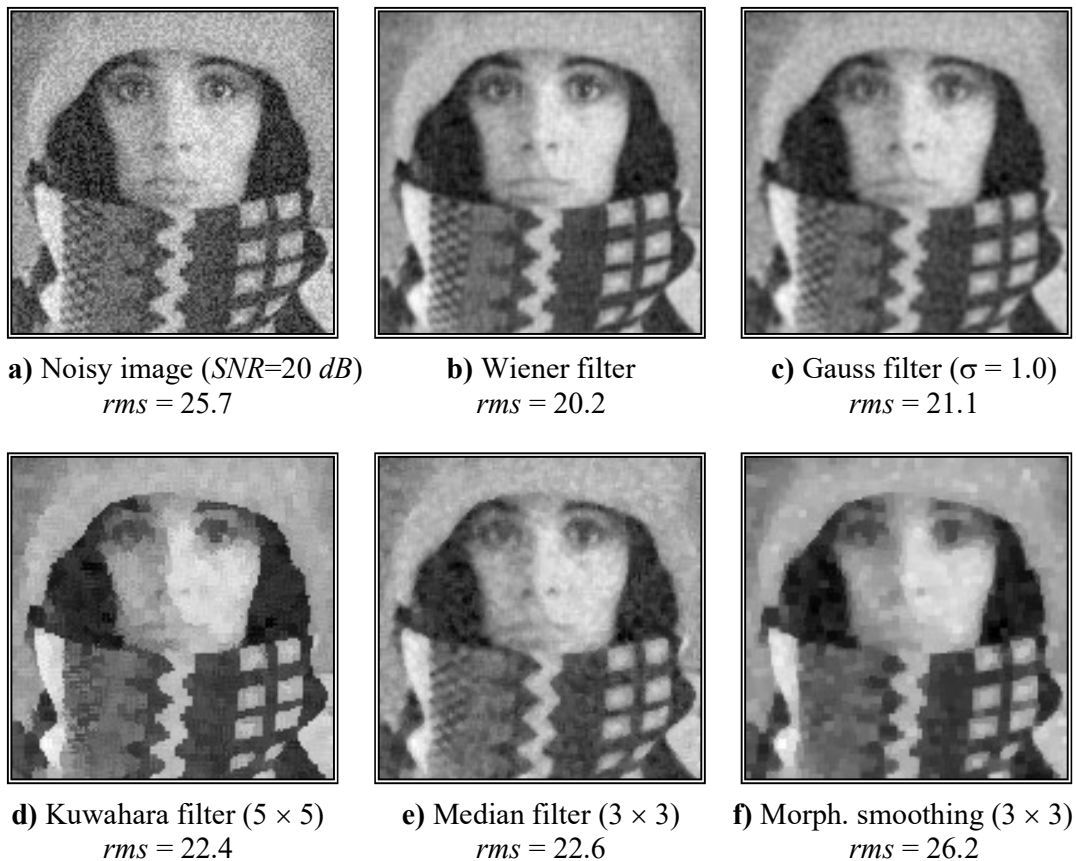


Figure 49: Noise suppression using various filtering techniques.

The root mean-square errors (*rms*) associated with the various filters are shown in Figure 49. For this specific comparison, the Wiener filter generates a lower error

than any of the other procedures that are examined here. The two linear procedures, Wiener filtering and Gaussian filtering, performed slightly better than the three non-linear alternatives.

Distortion suppression

The model presented above—an image distorted solely by noise—is not, in general, sophisticated enough to describe the true nature of distortion in a digital image. A more realistic model includes not only the noise but also a model for the distortion induced by lenses, finite apertures, possible motion of the camera and/or an object, and so forth. One frequently used model is of an image $a[m,n]$ distorted by a linear, shift-invariant system $h_o[m,n]$ (such as a lens) and then contaminated by noise $\kappa[m,n]$. Various aspects of $h_o[m,n]$ and $\kappa[m,n]$ have been discussed in earlier sections. The most common combination of these is the additive model:

$$c[m,n] = (a[m,n] \otimes h_o[m,n]) + \kappa[m,n] \quad (195)$$

The restoration procedure that is based on linear filtering coupled to a minimum mean-square error criterion again produces a Wiener filter [2]:

$$\begin{aligned} H_W(u, v) &= \frac{H_o^*(u, v) S_{aa}(u, v)}{|H_o(u, v)|^2 S_{aa}(u, v) + S_{nn}(u, v)} \\ &= \frac{H_o^*(u, v)}{|H_o(u, v)|^2 + \frac{S_{nn}(u, v)}{S_{aa}(u, v)}} \end{aligned} \quad (196)$$

Once again $S_{aa}(u,v)$ is the power spectral density of an image, $S_{nn}(u,v)$ is the power spectral density of the noise, and $H_o(u,v) = F\{h_o[m,n]\}$. Examination of this formula for some extreme cases can be useful. For those frequencies where $S_{aa}(u,v) \gg S_{nn}(u,v)$, where the signal spectrum dominates the noise spectrum, the Wiener filter is given by $1/H_o(u,v)$, the *inverse filter* solution. For those frequencies where $S_{aa}(u,v) \ll S_{nn}(u,v)$, where the noise spectrum dominates the signal spectrum, the Wiener filter is proportional to $H_o^*(u,v)$, the *matched filter* solution. For those frequencies where $H_o(u,v) = 0$, the Wiener filter $H_W(u,v) = 0$ preventing overflow.

The Wiener filter is a solution to the restoration problem based upon the hypothesized use of a linear filter and the minimum mean-square (or *rms*) error criterion. In the example below the image $a[m,n]$ was distorted by a bandpass

filter and then white noise was added to achieve an $SNR = 30 \text{ dB}$. The results are shown in Figure 50.

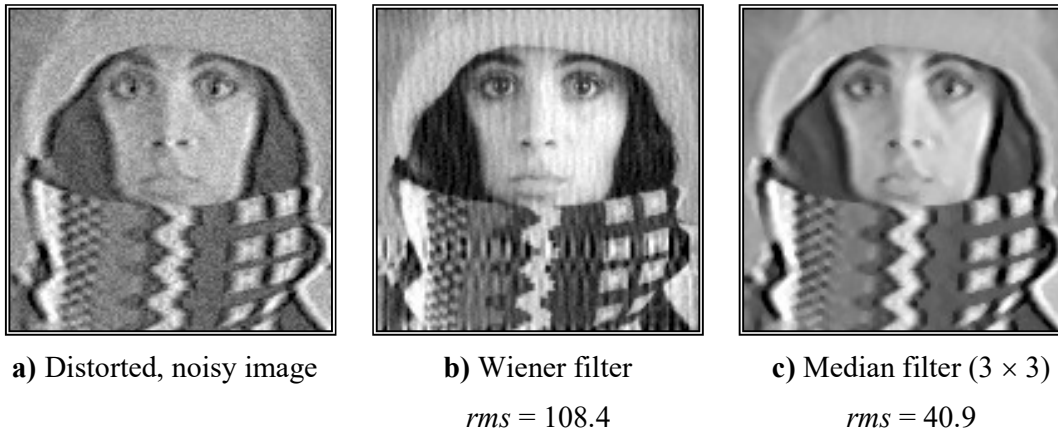


Figure 50: Noise and distortion suppression using the Wiener filter, eq. (196) and the median filter.

The rms after Wiener filtering but before contrast stretching was 108.4; after contrast stretching with eq. (77) the final result as shown in Figure 50b has a mean-square error of 27.8. Using a 3×3 median filter as shown in Figure 50c leads to a rms error of 40.9 before contrast stretching and 35.1 after contrast stretching. Although the Wiener filter gives the minimum rms error over the set of all linear filters, the non-linear median filter gives a lower rms error. The operation *contrast stretching* is itself a non-linear operation. The “visual quality” of the median filtering result is comparable to the Wiener filtering result. This is due in part to periodic artifacts introduced by the linear filter which are visible in Figure 50b.

SEGMENTATION

In the analysis of the objects in images it is essential that we can distinguish between the objects of interest and “the rest.” This latter group is also referred to as the background. The techniques that are used to find the objects of interest are usually referred to as *segmentation techniques* – segmenting the foreground from background. In this section we will two of the most common techniques—*thresholding* and *edge finding*— and we will present techniques for improving the quality of the segmentation result. It is important to understand that:

- there is no universally applicable segmentation technique that will work for all images, and,
- no segmentation technique is perfect.

Thresholding

This technique is based upon a simple concept. A parameter θ called the *brightness threshold* is chosen and applied to the image $a[m,n]$ as follows:

$$\begin{array}{ll} \text{If } a[m,n] \geq \theta & a[m,n] = \text{object} = 1 \\ \text{Else} & a[m,n] = \text{background} = 0 \end{array} \quad (197)$$

This version of the algorithm assumes that we are interested in light objects on a dark background. For dark objects on a light background we would use:

$$\begin{array}{ll} \text{If } a[m,n] < \theta & a[m,n] = \text{object} = 1 \\ \text{Else} & a[m,n] = \text{background} = 0 \end{array} \quad (198)$$

The output is the label “object” or “background” which, due to its dichotomous nature, can be represented as a Boolean variable “1” or “0”. In principle, the test condition could be based upon some other property than simple brightness (for example, $\text{If } (\text{Redness}\{a[m,n]\} \geq \theta_{\text{red}})$, but the concept is clear.

The central question in thresholding then becomes: How do we choose the threshold θ ? While there is no universal procedure for threshold selection that is guaranteed to work on all images, there are a variety of alternatives.

- *Fixed threshold* – One alternative is to use a threshold that is chosen independently of the image data. If it is known that one is dealing with very high-contrast images where the objects are very dark and the background is homogeneous (Section 10.1) and very light, then a constant threshold of 128 on a scale of 0 to 255 might be sufficiently accurate. By accuracy we mean that the number of falsely-classified pixels should be kept to a minimum.

- *Histogram-derived thresholds* – In most cases the threshold is chosen from the brightness histogram of the region or image that we wish to segment. (See Sections 3.5.2 and 9.1.) An image and its associated brightness histogram are shown in Figure 51.

A variety of techniques have been devised to automatically choose a threshold starting from the gray-value histogram, $\{h[b] \mid b = 0, 1, \dots, 2^B-1\}$. Some of the most common ones are presented below. Many of these algorithms can benefit from a smoothing of the raw histogram data to remove small fluctuations but the smoothing algorithm must not shift the peak positions. This translates into a zero-phase smoothing algorithm given below where typical values for W are 3 or 5:

$$h_{smooth}[b] = \frac{1}{W} \sum_{w=-(W-1)/2}^{(W-1)/2} h_{raw}[b-w] \quad W \text{ odd} \quad (199)$$

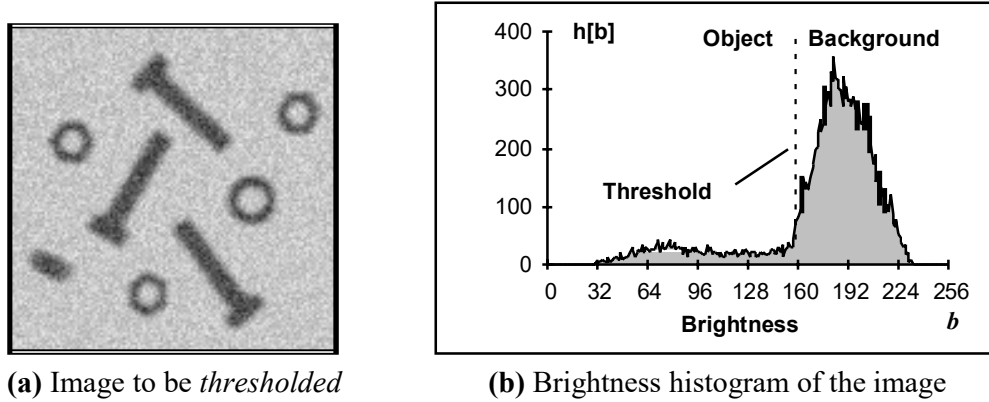


Figure 51: Pixels below the threshold ($a[m,n] < \theta$) will be labeled as object pixels; those above the threshold will be labeled as background pixels.

• *Isodata algorithm* – This iterative technique for choosing a threshold was developed by Ridler and Calvard [35]. The histogram is initially segmented into two parts using a starting threshold value such as $\theta_0 = 2^{B-1}$, half the maximum dynamic range. The sample mean ($m_{f,0}$) of the gray values associated with the foreground pixels and the sample mean ($m_{b,0}$) of the gray values associated with the background pixels are computed. A new threshold value θ_1 is now computed as the average of these two sample means. The process is repeated, based upon the new threshold, until the threshold value does not change any more. In formula:

$$\theta_k = (m_{f,k-1} + m_{b,k-1}) / 2 \text{ until } \theta_k = \theta_{k-1} \quad (200)$$

• *Background-symmetry algorithm* – This technique assumes a distinct and dominant peak for the background that is symmetric about its maximum. The technique can benefit from smoothing as described in eq. (199). The maximum peak ($maxp$) is found by searching for the maximum value in the histogram. The algorithm then searches on the *non-object pixel side* of that maximum to find a $p\%$ point as in eq. (39).

In Figure 51b, where the object pixels are located to the *left* of the background peak at brightness 183, this means searching to the right of that peak to locate, as an example, the 95% value. At this brightness value, 5% of the pixels lie to the *right* (are above) that value. This occurs at brightness 216 in Figure 51b. Because of the assumed symmetry, we use as a threshold a displacement to the *left* of the maximum that is equal to the displacement to the right where the $p\%$ is found. For

Figure 51b this means a threshold value given by $183 - (216 - 183) = 150$. In formula:

$$\theta = maxp - (p\% - maxp) \quad (201)$$

This technique can be adapted easily to the case where we have light objects on a dark, dominant background. Further, it can be used if the object peak dominates and we have reason to assume that the brightness distribution around the object peak is symmetric. An additional variation on this symmetry theme is to use an estimate of the sample standard deviation (s in eq. (37)) based on one side of the dominant peak and then use a threshold based on $\theta = maxp \pm 1.96s$ (at the 5% level) or $\theta = maxp \pm 2.57s$ (at the 1% level). The choice of “+” or “-” depends on which direction from $maxp$ is being defined as the object/background threshold. Should the distributions be approximately Gaussian around $maxp$, then the values 1.96 and 2.57 will, in fact, correspond to the 5% and 1 % level.

• *Triangle algorithm* – This technique due to Zack [36] is illustrated in Figure 52. A line is constructed between the maximum of the histogram at brightness b_{max} and the lowest value $b_{min} = (p=0)\%$ in the image. The distance d between the line and the histogram $h[b]$ is computed for all values of b from $b = b_{min}$ to $b = b_{max}$. The brightness value b_o where the distance between $h[b_o]$ and the line is maximal is the threshold value, that is, $\theta = b_o$. This technique is particularly effective when the object pixels produce a weak peak in the histogram.

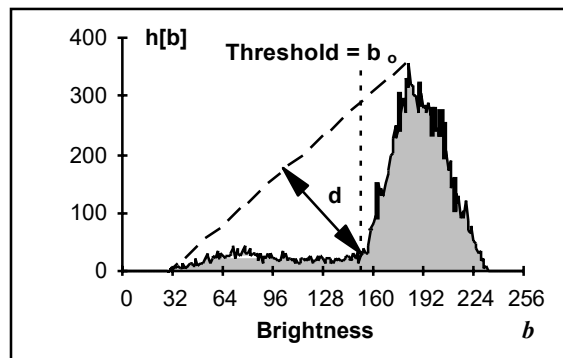


Figure 52: The triangle algorithm is based on finding the value of b that gives the maximum distance d .

The three procedures described above give the values $\theta = 139$ for the Isodata algorithm, $\theta = 150$ for the background symmetry algorithm at the 5% level, and $\theta = 152$ for the triangle algorithm for the image in Figure 51a.

Thresholding does not have to be applied to entire images but can be used on a region by region basis. Chow and Kaneko [37] developed a variation in which the

$M \times N$ image is divided into non-overlapping regions. In each region a threshold is calculated and the resulting threshold values are put together (interpolated) to form a thresholding surface for the entire image. The regions should be of “reasonable” size so that there are a sufficient number of pixels in each region to make an estimate of the histogram and the threshold. The utility of this procedure—like so many others—depends on the application at hand.

Edge finding

Thresholding produces a segmentation that yields all the pixels that, in principle, belong to the object or objects of interest in an image. An alternative to this is to find those pixels that belong to the borders of the objects. Techniques that are directed to this goal are termed *edge finding techniques*. From our discussion in Section 9.6 on mathematical morphology, specifically eqs. (79), (163), and (170), we see that there is an intimate relationship between edges and regions.

- *Gradient-based procedure* – The central challenge to edge finding techniques is to find procedures that produce *closed* contours around the objects of interest. For objects of particularly high *SNR*, this can be achieved by calculating the gradient and then using a suitable threshold. This is illustrated in Figure 53.

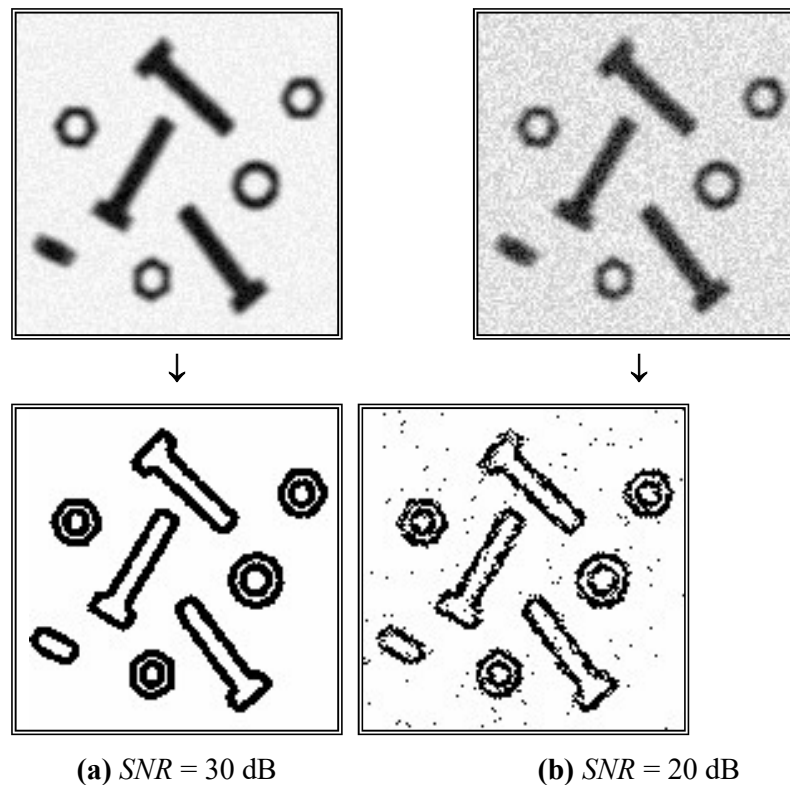


Figure 53: Edge finding based on the Sobel gradient, eq. (110), combined with the Isodata thresholding algorithm eq. (92).

While the technique works well for the 30 dB image in Figure 53a, it fails to provide an accurate determination of those pixels associated with the object edges for the 20 dB image in Figure 53b. A variety of smoothing techniques as described in Section 9.4 and in eq. (181) can be used to reduce the noise effects before the gradient operator is applied.

- *Zero-crossing based procedure* – A more modern view to handling the problem of edges in noisy images is to use the zero crossings generated in the Laplacian of an image (Section 9.5.2). The rationale starts from the model of an ideal edge, a step function, that has been blurred by an *OTF* such as Table 4 T.3 (out-of-focus), T.5 (diffraction-limited), or T.6 (general model) to produce the result shown in Figure 54.

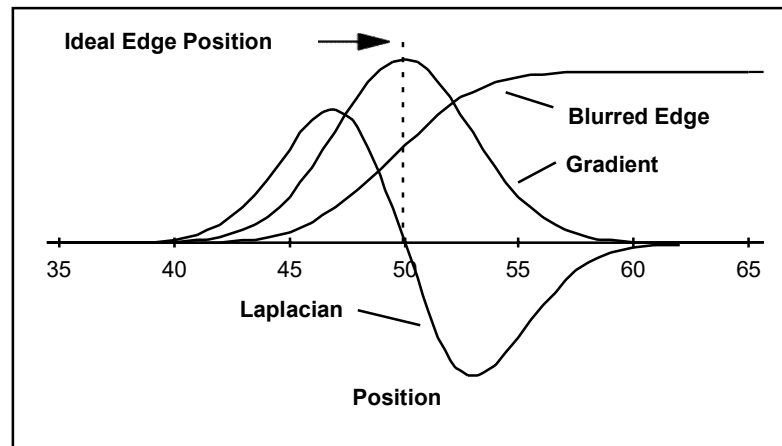


Figure 54: Edge finding based on the zero crossing as determined by the second derivative, the Laplacian. The curves are not to scale.

The edge location is, according to the model, at that place in the image where the Laplacian changes sign, the zero crossing. As the Laplacian operation involves a second derivative, this means a potential enhancement of noise in the image at high spatial frequencies; see eq. (114). To prevent enhanced noise from dominating the search for zero crossings, a smoothing is necessary.

The appropriate smoothing filter, from among the many possibilities described in Section 9.4, should according to Canny [38] have the following properties:

- In the frequency domain, (u,v) or (\wedge,Ψ) , the filter should be as narrow as possible to provide suppression of high frequency noise, and;
- In the spatial domain, (x,y) or $[m,n]$, the filter should be as narrow as possible to provide good localization of the edge. A too wide filter generates uncertainty as to precisely where, within the filter width, the edge is located.

The smoothing filter that simultaneously satisfies both these properties—minimum bandwidth and minimum spatial width—is the Gaussian filter described in Section 9.4. This means that the image should be smoothed with a Gaussian of an appropriate σ followed by application of the Laplacian. In formula:

$$\text{ZeroCrossing}\{a(x,y)\} = \left\{ (x,y) \mid \nabla^2 \{g_{2D}(x,y) \otimes a(x,y)\} = 0 \right\} \quad (202)$$

where $g_{2D}(x,y)$ is defined in eq. (93). The derivative operation is linear and shift-invariant as defined in eqs. (85) and (86). This means that the order of the operators can be exchanged (eq. (4)) or combined into one single filter (eq. (5)). This second approach leads to the Marr-Hildreth formulation of the “Laplacian-of-Gaussians” (*LoG*) filter [39]:

$$\text{ZeroCrossing}\{a(x,y)\} = \left\{ (x,y) \mid \text{LoG}(x,y) \otimes a(x,y) = 0 \right\} \quad (203)$$

where

$$\text{LoG}(x,y) = \frac{x^2 + y^2}{\sigma^4} g_{2D}(x,y) - \frac{2}{\sigma^2} g_{2D}(x,y) \quad (204)$$

Given the circular symmetry this can also be written as:

$$\text{LoG}(r) = \frac{r^2 - 2\sigma^2}{\sigma^4} e^{-(r^2/2\sigma^2)} \quad (205)$$

This two-dimensional convolution kernel, which is sometimes referred to as a “Mexican hat filter”, is illustrated in Figure 55.

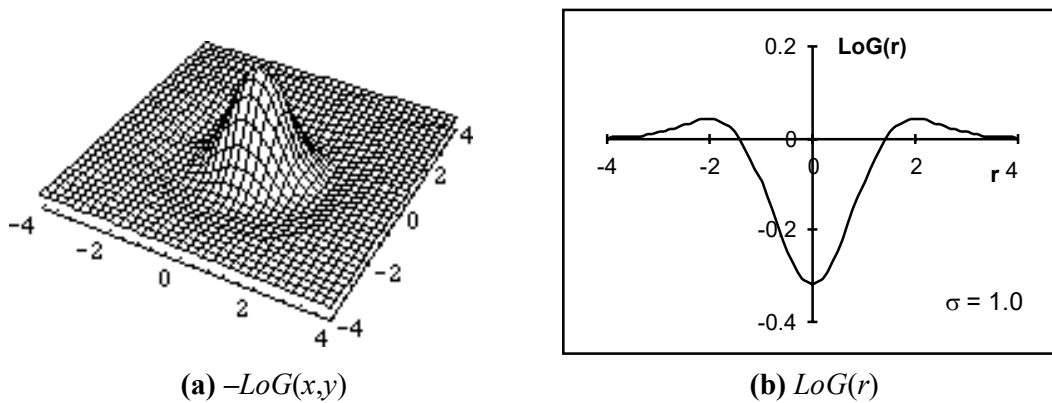


Figure 55: *LoG* filter with $\sigma = 1.0$.

•*PLUS-based procedure* – Among the zero crossing procedures for edge detection, perhaps the most accurate is the *PLUS* filter as developed by Verbeek and Van Vliet [40]. The filter is defined, using eqs. (121) and (122), as:

$$PLUS(a) = SDGD(a) + Laplace(a) \quad (206)$$

$$= \frac{\begin{bmatrix} & & & \\ & & & \\ & & & \\ & & & \end{bmatrix} A \begin{bmatrix} & & & \\ & & & \\ & & & \\ & & & \end{bmatrix} A^2 + 2 \begin{bmatrix} & & & \\ & & & \\ & & & \\ & & & \end{bmatrix} A \begin{bmatrix} & & & \\ & & & \\ & & & \\ & & & \end{bmatrix} A \begin{bmatrix} & & & \\ & & & \\ & & & \\ & & & \end{bmatrix} A^2}{A_x^2 + A_y^2} + \left(A_{xx} + A_{yy} \right)$$

Neither the derivation of the *PLUS*'s properties nor an evaluation of its accuracy are within the scope of this section. Suffice it to say that, for positively curved edges in gray value images, the Laplacian-based zero crossing procedure *overestimates* the position of the edge and the *SDGD*-based procedure *underestimates* the position. This is true in both two-dimensional and three-dimensional images with an error on the order of $(\sigma/R)^2$ where R is the radius of curvature of the edge. The *PLUS* operator has an error on the order of $(\sigma/R)^4$ if the image is sampled at, at least, $3\times$ the usual Nyquist sampling frequency as in eq. (56) or if we choose $\sigma \geq 2.7$ and sample at the usual Nyquist frequency.

All of the methods based on zero crossings in the Laplacian must be able to distinguish between zero *crossings* and zero *values*. While the former represent edge positions, the latter can be generated by regions that are no more complex than bilinear surfaces, that is, $a(x,y) = a_0 + a_1 \bullet x + a_2 \bullet y + a_3 \bullet x \bullet y$. To distinguish between these two situations, we first find the zero crossing positions and label them as "1" and all other pixels as "0". We then multiply the resulting image by a measure of the *edge strength* at each pixel. There are various measures for the edge strength that are all based on the gradient as described in Section 9.5.1 and eq. (182). This last possibility, use of a morphological gradient as an edge strength measure, was first described by Lee, Haralick, and Shapiro [41] and is particularly effective. After multiplication the image is then thresholded (as above) to produce the final result. The procedure is thus as follows [42]:

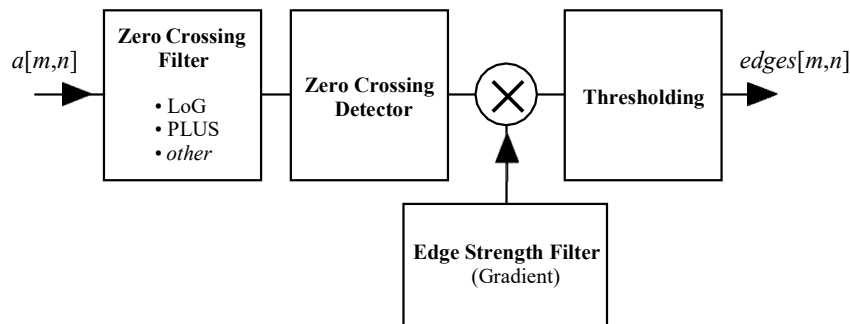


Figure 56: General strategy for edges based on zero crossings.

The results of these two edge finding techniques based on zero crossings, *LoG* filtering and *PLUS* filtering, are shown in Figure 57 for images with a 20 dB SNR.

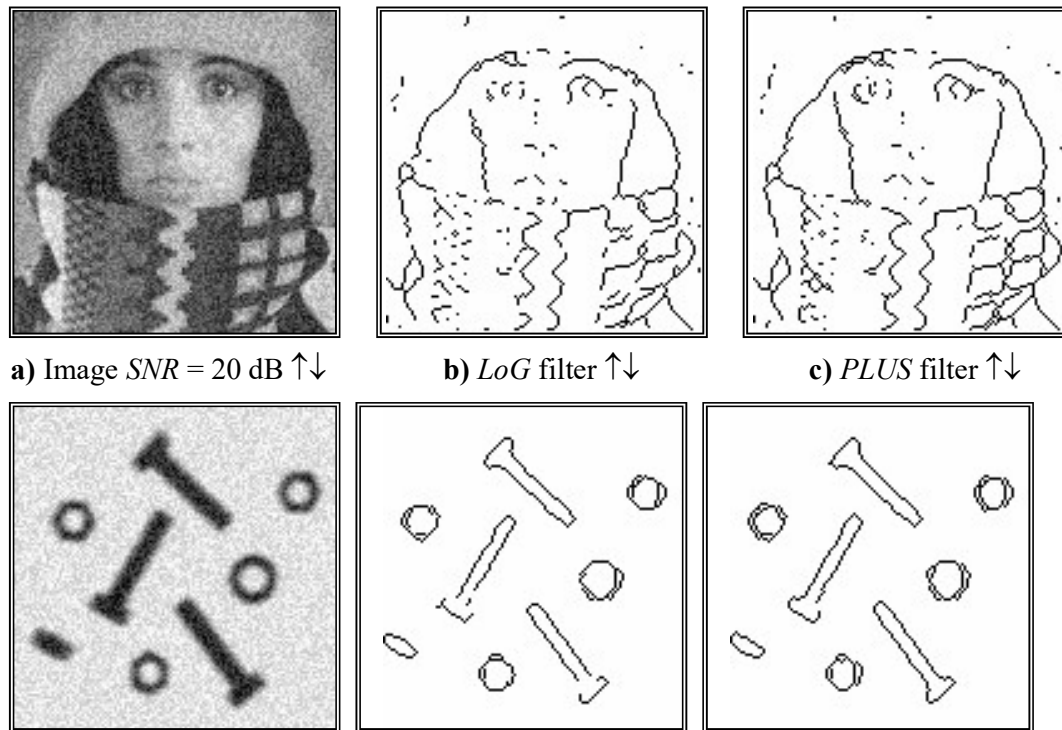


Figure 57: Edge finding using zero crossing algorithms *LoG* and *PLUS*. In both algorithms $\sigma = 1.5$.

Edge finding techniques provide, as the name suggests, an image that contains a collection of edge pixels. Should the edge pixels correspond to objects, as opposed to say simple lines in the image, then a region-filling technique such as eq. (170) may be required to provide the complete objects.

DATA ANALYSIS

Companion audio and video recordings were synchronized and loaded into a custom graphic user interface for inspection and analysis (Proctor et al., 2010a; Narayanan et al., 2011), so that MR image sequences could be examined to determine the mechanisms of production of each of the sound effects in the subject's repertoire.

Start and end times delineating each token were identified by examining the audio signal, spectrogram, and time-aligned video frames, and the corresponding intervals of each signal were labeled. Laryngeal displacement was calculated by manually locating the end points of the glottal trajectory using a measurement cursor superimposed on the video frames. The coordination of glottal and supraglottal gestures was examined to provide insights into the airstream

mechanisms exploited by the artist to produce different effects.

Beatboxing grooves produced by the subject were manually transcribed. Using MuseScore (v1.2) musical notation software, the proposed transcriptions were encoded in MIDI format, exported as WAV audio, and compared to the audio recordings of the corresponding performance segment. To ensure that the annotated percussion sequences captured the musical properties of the grooves performed by the subject as accurately as possible, the musical scores and specifications for percussion ensemble, tempo and dynamics were adjusted, along with the MIDI sound palates, until the synthesized audio closely approximated the original recordings.

METHODS

Simulation of Spatial Versus Time Resolution Trade-offs With Spiral Sampling
A multishot short spiral readout spoiled gradient echo pulse sequence (flip angle //FA: 15 degrees; slice FIG. 1. Spatial versus temporal resolutions trade-offs in RT-MRI using short interleaved spiral trajectories. In comparison to full sampling, sparse sampling reduces spatiotemporal trade-offs and enables improved visualization of several speech tasks both in single- and multiplane imaging. The clouds in the above figure represent a recently reported consensus opinion among speech imaging researchers and linguists (3). 2 Lingala et al. thickness: 6 mm; readout time: 2.5 ms, repetition time [TR] $\frac{1}{4}$ 6.004 ms), which was used in our previous studies, was adapted in this study (32,33). We chose spiral trajectories over alternate trajectories because they have shown to provide a superior trade-off among spatial resolution, time resolution, and robustness to motion artifacts. The spiral trajectories were designed to make maximum use of gradients (40 mT/m maximum gradient amplitude and 150 mT/m/ms slew rate). Simulations were performed to investigate the spatial versus time resolution trade-offs for a field of view (FOV) of 20 cm² at Nyquist (full) sampling and rate 6.5-fold undersampling for single-slice and concurrent three-slice imaging (Figure 1). Nyquist sampling was determined by varying the number of spiral interleaves such that the maximum spacing between interleaves equaled the reciprocal of the unaliased FOV. The Nyquist definition was defined based on the assumption of spiral interleaving with a uniform angle distribution. Figure 1 also duplicates the schematic placement of various speech tasks according to their spatial and temporal resolution requirements, as reported in Lingala et al (3). Single-Slice and Multislice Golden Angle Spiral Time Interleaving A previously proposed golden angle time interleaved sampling pattern scheme in which successive spiral interleaves are separated by the golden angle $2\pi/(5 \pm 1)$ was adapted in this work (32). The sampling schedule was repeated after 144 interleaves. Two single-slice sequences with spatial resolutions of 2.4 and 1.76 mm² were realized, which respectively corresponded to 13 and 21 spiral interleaves/frame for Nyquist sampling. A flexible multislice time interleaved sequence (33) was also adapted to realize an arbitrary three-slice select sequence at 2.4 mm². The golden angle

increment for the three-slice sequence occurred every 3 TRs. It should be noted that the unaliased FOV with golden angle sampling slightly differs with that of uniform density sampling used for the simulation in Figure 1 (32). Specifically, the point spread function of golden angle spiral sampling provides more reduced side-lobe energies and provides improved tradeoff of the achievable unaliased FOV in comparison to uniform density sampling (32).

Custom Upper Airway Coil

All of our experiments were performed on a GE Signa Excite 1.5T scanner (GE Healthcare, Little Chalfont, UK) with a custom eight-channel upper airway receiver coil that has four elements on either side of the jaw. The elements were designed to be spatially localized and to be in close proximity to the upper airway to provide high SNR over all the important upper airway structures. We chose custom coil design because of its ability to provide superior SNR across all articulators of interest. This is advantageous because it provides an important boost in SNR while operating at 1.5T, and enables efficient combination with spirals is highly complementary, because it together enables improved SNR at low fields, low off-resonance-related spiral artifacts, and high sampling efficiency. The custom coil was developed for two sizes for adult and child arrays, although all the experiments in the current study were performed on adults using the adult-sized array. Reconstruction On-the-Fly Data acquisition was implemented within custom RThawk software (34). A view-sharing scheme was used, where data for each frame were combined respectively from 13 and 21 subsequent interleaves, respectively for the 2.4 and 1.76 mm² sequences. Images were reconstructed on-the-fly by using a fast implementation of the gridding algorithm within RT-hawk. The minimal latency allowed for instant feedback to the operator and enabled efficient scan plane localizations. In addition, it enabled on-the-fly adjustment of the center frequency to minimize off-resonance blurs. Specifically, the subject being scanned was asked to open their mouth, and in the midsagittal plane, the operator qualitatively adjusted the center frequency such that the air-tissue (majorly: air-tongue, air-velum, air-lip) boundaries were sharp. Offline A sparse SENSE temporal finite difference constrained reconstruction scheme was implemented offline. This constraint exploits redundancy based on the fact that the desired information is contained in the moving edges. This directly fits to the application of speech imaging, where the end goal is the assessment of interaction and timing of various articulators, or assessment of the dynamics of the air-tissue interfaces (moving edges). Spatial and temporal finite difference constraints have also been previously used in several studies (e.g., (35–39)). The reconstruction is formulated as shown by Equation [1]: $\min_{\mathbf{f}(\mathbf{x},\mathbf{t})} \|\mathbf{b} - \mathbf{A}\mathbf{f}(\mathbf{x},\mathbf{t})\|_2$ [1] where \mathbf{b} is a concatenated vector containing the spiral noisy k-t measurements from each coil and $\mathbf{f}(\mathbf{x},\mathbf{t})$ is the dynamic data to be reconstructed at a retrospectively specified time resolution. A model coil sensitivity encoding as well as Fourier encoding on the specified spiral trajectory in each time frame; the coil sensitivities were assumed to be time-invariant, and were estimated by an Eigen decomposition method using time-averaged image data from each coil (40), and the nonuniform Fourier transform (nuFFT) implementation by Fessler and

Sutton (41) was used. The l_1 -sparsity-based temporal finite difference (Dt) penalty is used to penalize pixels with rapidly varying pixel time profiles. λ is the regularization parameter that controls the balance between the sparsity penalty and the data fidelity. Equation [1] was solved by a nonlinear conjugate gradient (CG) algorithm (42). The algorithm was initialized with the f5AH(b) estimate and was terminated at 40 iterations, where qualitatively there was no noticeable change in image quality. The reconstructions were implemented within MATLAB (The MathWorks, Inc., Natick, MA) on an Intel Core i7 3.5 GHz machine with 32-GB memory. A Fast and Flexible MRI system to study vocal tract shaping 3 In Vivo Experiments and Speech Tasks Three healthy volunteers (2 male, 1 female; median age: 29) and 1 male tongue cancer patient (62 years) were scanned. The patient was scanned before clinical treatment. All the stimuli were presented in the scanner using a mirror projector setup. A variety of speech tasks were considered. The midsagittal orientation was used for single-slice sequences, whereas orientations for the multislice sequence differed according to the speech task. With the 2.4-mm² single-slice sequence, volunteer 1 (male Indian English speaker) was scanned without any speech stimuli on two separate instances: (a) using an eight-channel head coil and (b) using the custom eight-channel upper airway coil. With the custom upper airway coil, the same volunteer was scanned with both the single-slice sequences, using the stimuli: “one-twothree-four-five” at a normal speech rate followed by a rapid speech rate (approximately 4 times faster). A task to produce interleaved consonant and vowel sounds by the repetition of the phrase: “loo-lee-laa-za-na-za” at the normal speech rate was considered on volunteer 1 and imaged using the three-slice sequence (one midsagittal, one axial plane at the level of mid-pharyngeal airway, and one coronal plane at the middle of the tongue). Volunteer 2 (male Chinese English speaker) was scanned with the 1.76-mm² single-slice sequence, while producing the sentence: “She had your dark suit in greasy wash water,” which involves producing sounds that involve rapid articulatory movements (e.g., coarticulation events as a part of running speech). Volunteer 3 (female American English speaker) was a beat boxer and was scanned while producing a variety of beat-boxing sounds. The 2.4-mm² single- and concurrent three-slice (one midsagittal slice, one axial slice at the level of velum, and one axial slice at the level of glottis) sequences were considered. The particular axial cuts were chosen to capture the rapid velar and glottis movements during beat boxing. Volunteer 4 (male American English speaker) was a tongue cancer patient and was scanned with the single-slice 2.4-mm² sequence. Speech stimuli comprising words, sentences, and a passage were presented, and the ability to produce speech was analyzed. A small subset of these stimuli is presented in this work, which pertain to short words that contain vowels interleaved by consonants: “beat, bit, bait, bet, bat, pot, bought, boat.” Simultaneous Audio Collection For 3 of 4 volunteers scanned, audio recordings were obtained simultaneously at a sampling frequency of 20 KHz inside the scanner, while the subjects were being imaged, using a commercial fiber optic microphone (Optoacoustics Ltd., Or Yehuda, Israel) (43)

and a custom recording setup. Noise cancellation was performed using a data-driven adaptive signal processing algorithm, which is blind to the acoustic properties of noise (44). The final noise-cancelled audio was synchronized with the reconstructed RT-MRI data to facilitate acousticarticulatory analysis.

Analysis Comparison of SNR Between Coils

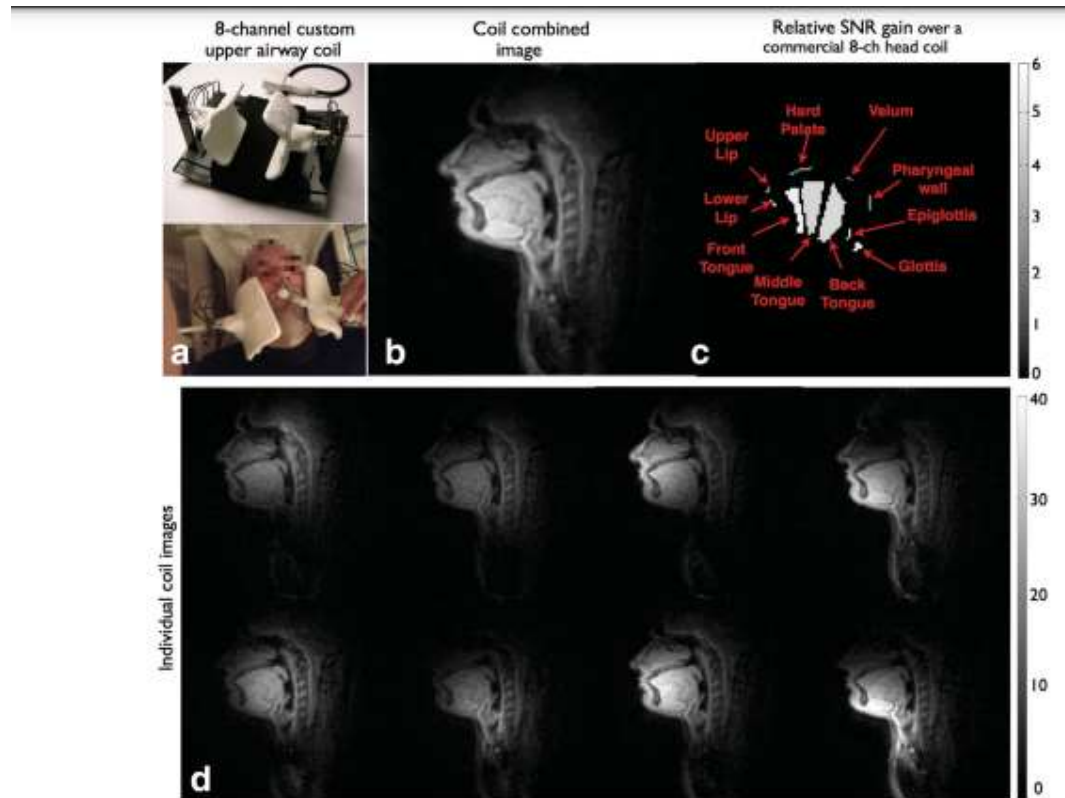
The SNR properties of the custom eight-channel upper airway coil were qualitatively compared with a commercial eight-channel head (brain) coil. The single-slice (2.4 mm²) sequence was used, where volunteer 1 was scanned with no speech, and 55 interleaves were used to reconstruct f5AH(b). The ROI SNR in different upper airway regions were quantified as $SNR_{ROI} = \frac{1}{n} \sum_{i=1}^n \frac{S_i}{s_i}$, where S is a vector with image intensities from the ROI containing a specific upper airway structure, and n is a vector with intensities from an ROI in the background capturing only noise. A total of 10 ROIs were defined: 1) upper lip; 2) lower lip; 3) front tongue; 4) middle tongue; 5) back tongue; 6) hard palate; 7) velum; 8) pharyngeal wall; 9) epiglottis; and 10) glottis. A relative measure of SNR between the two coils was evaluated by the factor SNR_{UA}/SNR_{head} .

Choosing Regularization Parameter in Constrained Reconstruction

The regularization parameter in the constrained reconstruction was chosen empirically, with the best trade-off between artifact removal, and temporal stair-casing. With the single-slice 2.4-mm² sequence, and a 12-ms time resolution reconstruction, the effect of regularization parameter on the reconstructions was studied. L-curves were obtained for two data sets with different speech stimuli from different volunteers: volunteer 1 with fast speech stimuli of counting numbers and volunteer 3 with beat boxing. For generating the L-curve, the CG iterations were set to a very high number of 120 to ensure no bias in the norm calculations resulting from small numerical errors; however, in practice, a lower number of iterations of around 40 were sufficient for convergence, where, qualitatively, there was no noticeable change in image quality.

Comparison of Constrained Reconstructions at Various Reduction Factors

Golden-angle time interleaving allows for retrospective reconstruction with arbitrary time resolution. Using the speech stimuli of counting numbers at a rapid pace, reconstructions were performed at 5, 3, 2, and 1 TR for the 2.4- and 1.76-mm² single-slice sequences. This corresponded to reduction factors (R) between 2.6- and 13-fold for the 2.4-mm² sequence and 4.2- to 21-fold for the 1.76-mm² sequence. The regularization parameters were chosen empirically for all cases. A $12.4\text{mm}^2 \times 0.1$ and $11.76\text{mm}^2 \times 0.3$ respectively for the 2.4- and 1.76-mm² sequence was used for the 5, 3, and 2 TR cases, whereas a higher $12.4\text{mm}^2 \times 0.2$ and $11.76\text{mm}^2 \times 0.4$ was used for the 1 TR case. The trade-off between residual aliasing artifacts, and temporal blurring (resulting from large temporal footprint), and reconstruction time was analyzed in all the cases.



In the ongoing experiments, Patil and four other beatboxers of different ages, genders and skill levels took turns lying down in an MRI machine while demonstrating their repertoires of homemade percussive sounds. While the beatboxers filled the MRI with rhythmic strings of clicks, kicks, rattles and trills, the machine recorded the exact anatomical movements occurring inside their mouths, noses and throats. The result is a literal inside look at the mechanics of beatboxing, captured in high-definition video.

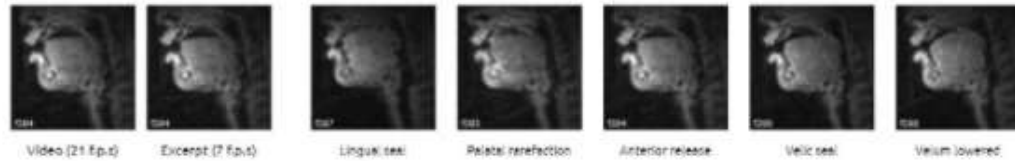
Timothy Greer, a doctoral candidate at USC and a member of the SPAN beatboxing team, said these videos counter previous research that suggested beatboxers can create only those sounds that fit into the phonetic library of known world languages. In fact, it looks more like beatboxers invent a new language all their own, he said.

"Beatboxers are able to mimic percussive sounds that we don't know to exist in any language," Greer told Live Science. "They're learning to use their mouths and vocal tracts in ways that they have never had to use for speech, going totally outside of common articulations and airstreams and creating what we call art. It's incredible."

Snare Drum Effects

Three different snare drum effects were demonstrated by the subject, each produced with different articulatory and airstream mechanisms: a click, an ejective affricate, and a pulmonic egressive dorsal stop-fricative sequence.

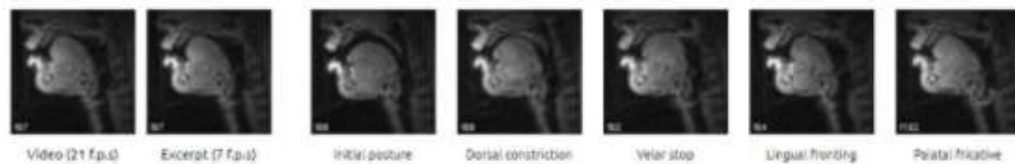
'Clap' snare [ŋ]



'No mesh' snare [pʰ:ʊ]



'Meshed' snare [kç:]



Using MRI recordings, the SAIL team is working on breaking down the precise vocal tract movements behind every single sound in a beatboxer's repertoire. Here are the moves behind three different snare effects. (Image credit: Timothy Greer, Signal Analysis Interpretation Laboratory)

For example, Greer said, watch a beatboxer perform an "inward click roll," a sound that roughly mimics a rattling synth bass drum. To articulate this sound, the beatboxer needs to curl her tongue back on itself while inhaling just enough air to cause a trilling vibration. According to Greer, the resulting sound comes from an airstream "we don't hear in any known language."

By stringing together percussive articulations like these into cohesive beats, beatboxers essentially organize sounds into longer "words and phrases," Greer said, much like what happens in speech. The difference is that there are no native speakers of beatboxing; it's a nonverbal language that comes entirely from mimicry and experimentation. Luckily, Greer said, that means anyone who wants to should be able to learn it.

A team of researchers is using real-time MRI to study the production of beatboxing sounds. Timothy Greer, a doctoral candidate at the University of Southern California, will describe their work showing how real-time MRI can characterize different beatboxing styles and how video signal processing can

demystify the mechanics of artistic style. Greer will present the study at the Acoustical Society of America's 176th Meeting, held in conjunction with the Canadian Acoustical Association's 2018 Acoustics Week in Canada, Nov. 5-9 at the Victoria Conference Centre in Victoria, Canada.

The team is interested in using real-time MRI data to observe the vocal tracts of beatboxers just before they make a sound to see if those movements are different from speech. The real-time MRI data provides a dynamic view of the entire midsagittal vocal tract and at a frame rate high enough to observe the movement and coordination of critical articulators.

"Beatboxers may learn something different in preparing to make a sound than they do when they're talking," said Greer. "Using real-time MRI allows us to investigate the difference in the production of music and language and to see how the mind parses these different modalities."

Previous research in this field usually consists of a case study of one beatboxer and suggests that beatboxers can only produce sounds that exist within the world's known languages. The new study looks at several beatboxers of different ages and genders, and with different native languages.

"We found that beatboxers can create sounds that are not seen in any language. They have an acrobatic ability to put together all these different sounds," said Greer. "They can hear a sound like a snare drum and they can figure out what they need to do with their mouth to re-create it."

One of the main challenges for the researchers has been developing the algorithms they use to quantify the movement of the beatboxer's vocal tract. A linguist labels all the various parts of the body involved in sound production such as the tongue and the roof of the mouth, and the algorithm tracks the images of these various parts as they move during the production of sound.

"The vocal tract is amazing but it's also incredibly complex. We need to keep creating better computer algorithms to understand how it all works together," said Greer.

This work is supported by NIH grant R01DC007124 and NSF grant IIS 1514544.

Presentation #3aMU5, "How beatboxers produce percussion sounds: A real-time magnetic resonance imaging investigation," by Timothy Greer, Shri Narayanan, Reed Blaylock and Nimisha Patil will take place Wednesday, Nov. 7, 11:25 a.m. in the Crystal Ballroom (FE) of the Victoria Conference Center in Victoria, British Columbia, Canada.

Bass/Kick Drum Effects

Kick drum effects were all produced as bilabial ejectives.

'Punchy' kick [pʰːʌ]



'Thud' kick [pʰːɪ]



Snare Drum Effects

Three different snare drum effects were demonstrated by the subject, each produced with different articulatory and airstream mechanisms: a click, an ejective affricate, and a pulmonic egressive dorsal stop-fricative sequence.

'Clap' snare [ŋ]



'No mesh' snare [pʰːʊ]



'Rim Shot' effect [kʰ]



'Rim Shot k' effect [kʰhː]



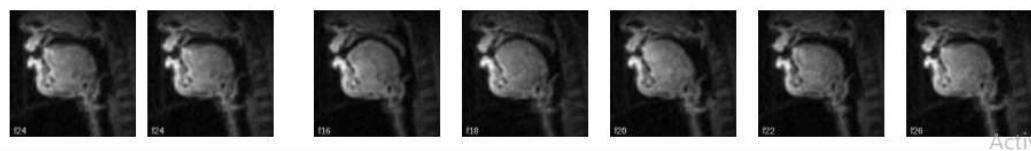
Hi-hat Effects

Five hi-hat effects were demonstrated by the subject, all produced as affricates of rapid stop-fricative sequences. One effect made use of glottalic ingressive (implosive) airstream mechanism.

'Open K' hi-hat [ks:]



'Open T' hi-hat [ts:]



Using the mouth, lips, tongue and voice to generate sounds that one might never expect to come from the human body is the specialty of the artists known as beatboxers. Now scientists have used scanners to peer into a beatboxer as he performed his craft to reveal the secrets of this mysterious art.

The human voice has long been used to generate percussion effects in many cultures, including North American scat singing, Celtic lilting and diddling, and Chinese kouji performances. In southern Indian classical music, konnakol is the percussive speech of the solkattu rhythmic form. In contemporary pop music, the relatively young vocal art form of beatboxing is an element of hip-hop culture.

Until now, the phonetics of these percussion effects were not examined in detail. For instance, it was unknown to what extent beatboxers produced sounds already used within human language.

To learn more about beatboxing, scientists analyzed a 27-year-old male performing in real-time using MRI. This gave researchers "an opportunity to study the sounds people produce in much greater detail than has previously been possible," said Shrikanth Narayanan, a speech and audio engineer at the University of Southern California in Los Angeles. "The overarching goals of our work drive at larger questions related to the nature of sound production and mental processing in human communication, and a study like this is a small part of the larger puzzle."

Related Multimedia: Watch and listen to beatboxing examples

The investigators made 40 recordings each lasting 20-40 seconds long as the beatboxer produced all the effects in his repertoire, as individual sounds, composite beats, rapped lyrics, sung lyrics and freestyle combinations of these elements. He categorized 17 distinct percussion sounds into five instrumental classes — kick drums, rim shots, snare drums, hi-hats, and cymbals. The artist demonstrated his repertoire at several different tempos, ranging from slower at roughly 88 beats per minute, to faster at 104.

"We were astonished by the complex elegance of the vocal movements and the sounds being created in beatboxing, which in itself is an amazing artistic display," Narayanan said. "This incredible vocal instrument and its many capabilities continue to amaze us, from the intricate choreography of the 'dance of the tongue' to the complex aerodynamics that work together to create a rich tapestry of sounds that encode not only meaning but also a wide range of emotions."

"It is absolutely amazing that a person can make these sounds — that a person has such control over the timing of various parts of the speech apparatus," said phonetician Donna Erickson at the Showa University of Music and Sophia University, both in Japan, who did not participate in this study. "It is very exciting to see how far technology has come — that we can see these movements in real time. It gives us a much better understanding of how the various parts of our speech anatomy work."

The data suggest that "the sounds used by our beatboxing artist mirror those found in the diverse sound systems of the world's many languages," said researcher Michael Proctor, a linguist and speech scientist at the University of Western Sydney in Australia.

The scientists found the beatboxer, a speaker of American English and Panamanian Spanish, was able to generate a wide range of sound effects that do not appear in either of the languages he spoke. Instead, they appeared similar to clicks seen in African languages such as Xhosa from South Africa, Khoekhoe from Botswana, and !Xóõ from Namibia, as well as ejective consonants — bursts of air generated by closing the vocal cords — seen in Nuxálk from British Columbia, Chechen from Chechnya and Hausa from Nigeria and other countries in Africa.

"A key finding of our work is to show that we can describe the basic sounds used by the artist with the same system used to describe speech sounds, which suggests that there is a common inventory of sounds that are drawn upon to create any vocal expression," Proctor said.

The research also sheds light on the human ability to emulate sounds, and on how the human instincts for music and language can overlap and converge. Also, "learning more about beatboxing and other forms of vocal musical expression may offer insights into novel future speech therapy," Narayanan said.

"It would be interesting to see if even more unusual sounds could be both imitated and incorporated," said speech scientist Doug Whalen at Yale University, who did not take part in this research. In addition, "it would be nice to know how the beatboxer came by his inventory, and how long it took him to find the articulations that satisfied him. Were they quickly found? Or quite difficult?"

One goal of future research is to image more of the tongue and palate to provide more details of the mechanics of beatboxing. "It is very humbling to realize that we still don't fully understand some of these fundamental human capabilities," Narayanan said.

In addition, further studies will examine other practitioners of vocal percussion. One goal is to explore how some beatboxers can create the illusion of multiple instruments, or make percussive noises while simultaneously humming or speaking.

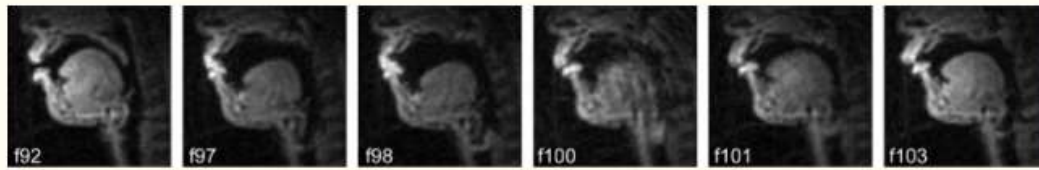


Figure 1

Articulation of a “punchy” kick drum effect as an affricated labial ejective [pfʰ.ɹ̥]. Frame 92: starting posture; f97: lingual lowering, velic closure; f98: fully lowered larynx, glottalic closure; f100: rapid laryngeal raising accompanied by lingual raising; f101: glottis remains closed during laryngeal raising; f103: glottal abduction; final lingual posture remains lowered.

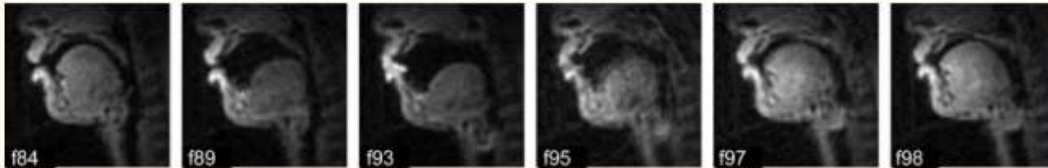


Figure 2

Articulation of a “thud” kick drum effect as a bilabial ejective [pʰ]. Frame 84: starting posture; f89: glottal lowering, lingual retraction; f93: fully lowered larynx, sealing of glottalic, velic and labial ports; f95: rapid laryngeal raising accompanied by lingual raising; f97: glottis remains closed during laryngeal raising and lingual advancement; f98: final lingual posture raised and advanced.

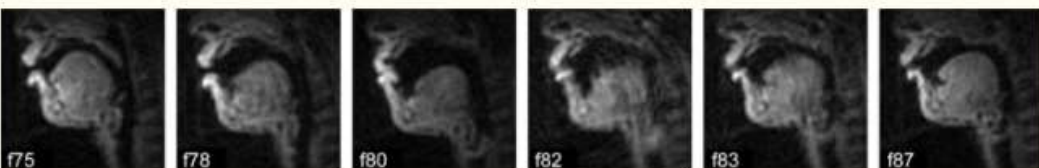


Figure 3

Articulation of an “808” kick drum effect as a bilabial ejective [pʰʊ]. Frame 75: starting posture; f78: lingual lowering, velic closure; f80: fully lowered larynx, glottalic and labial closure; f82: rapid laryngeal raising, with tongue remaining retracted; f83: glottis remains closed during laryngeal raising; f87: glottal abduction; final lingual posture midhigh and back.

The effect described as a “punchy kick” (SBN: bf) was produced as a bilabial affricate ejective /pfʰ.ɹ̥/. Six image frames acquired over a 550 ms interval during the production of one token are shown in Fig. Fig.1.1. Laryngeal lowering and lingual retraction commence approximately 350 ms before the acoustic release burst; labial approximation commences 230 ms before the burst. Velic raising to seal the nasopharynx off from the oral vocal tract can be observed as the larynx is lowered and the lips achieve closure (frame 97). Glottal

closure is clearly evident after the larynx achieves the lowest point of its trajectory (frame 98). Rapid upward movement of the larynx can be observed after glottal adduction, accompanied by rapid raising of the tongue dorsum, resulting in motion blurring throughout the posterior oral and supralaryngeal regions (frame 100).

Mean upward vertical displacement of the glottis during ejective production, measured over five repetitions of the punchy kick drum effect, was 21.0 mm. The glottis remained adducted throughout the production of the ejective (frame 101), and was reopened approximately 160 ms after the beginning of the acoustic release burst. At the completion of the ejective, the tongue remained in a low central position (frame 103) resembling the articulatory posture observed during the subject's production of the vowel [ʌ].²

In addition to the punchy kick, the subject controlled two variant bass drum effects (SBN: b), both produced as unaffricated bilabial ejective stops: a “thud kick,” and an “808 kick.” Image sequences acquired during production of these effects are shown in Figs. Figs.2223,3, respectively. The data reveal that although the same basic articulatory sequencing is used, there are minor differences in labial, glottal, and lingual articulation which distinguish each kick drum effect.

In both thud and 808 kick effects, the lips can be seen to form a bilabial seal (Fig. (Fig.2,2, frames 93–95; Fig. Fig.3,3, frames 80–82), while in the production of the affricated punchy effect, the closure is better characterized as labio-dental (Fig. (Fig.1,1, frames 98–103). Mean upward vertical displacement of the glottis during ejective production, measured over six repetitions of the thud kick drum effect, was 18.6 mm, and in five of the six tokens demonstrated, no glottal abduction was observed after completion of the ejective. Vertical glottal displacement averaged over five tokens of the 808 kick drum effect, was 17.4 mm. Mean duration (oral to glottal release) of the 808 effect was 152 ms.

A final important difference between the three types of kick drum effects produced by this subject concerns lingual articulation. Different amounts of lingual retraction can be observed during laryngeal lowering before production of each ejective. Comparison of the end frames of each image sequence reveals that each effect is produced with a different final lingual posture. These differences can be captured in close phonetic transcription by using unvoiced vowels to represent the final posture of each effect: [pʰ□□□:ʌ°](punchy), [pʰI°](thud), and [pʰʊ°] (808).

These data suggest that the kick drum effects produced by this artist are best characterized as “stiff” (rather than “slack”) ejectives, according to the typological classification developed by Lindau (1984), Wright et al. (2002), and Kingston (2005): all three effects are produced with a very long voice onset time (VOT), and a highly transient, high amplitude aspiration burst. The durations of these sound effects (152 to 160 ms) are longer than the durations reported for glottalic egressive stops in Tlingit (Maddieson et al., 2001) and Witsuwit'en (Wright et al., 2002), but resemble average release durations of some other Athabaskan glottalic consonants (Hogan, 1976; McDonough and Wood, 2008). In general, it appears that the patterns of coordination between glottal and oral closures in these effects more closely resemble those observed in North American languages, as opposed to African languages like Hausa (Lindau, 1984), where “the oral and glottal closures in an ejective stop are released very close together in time” (Maddieson et al., 2001).

Articulation of rim shot effects

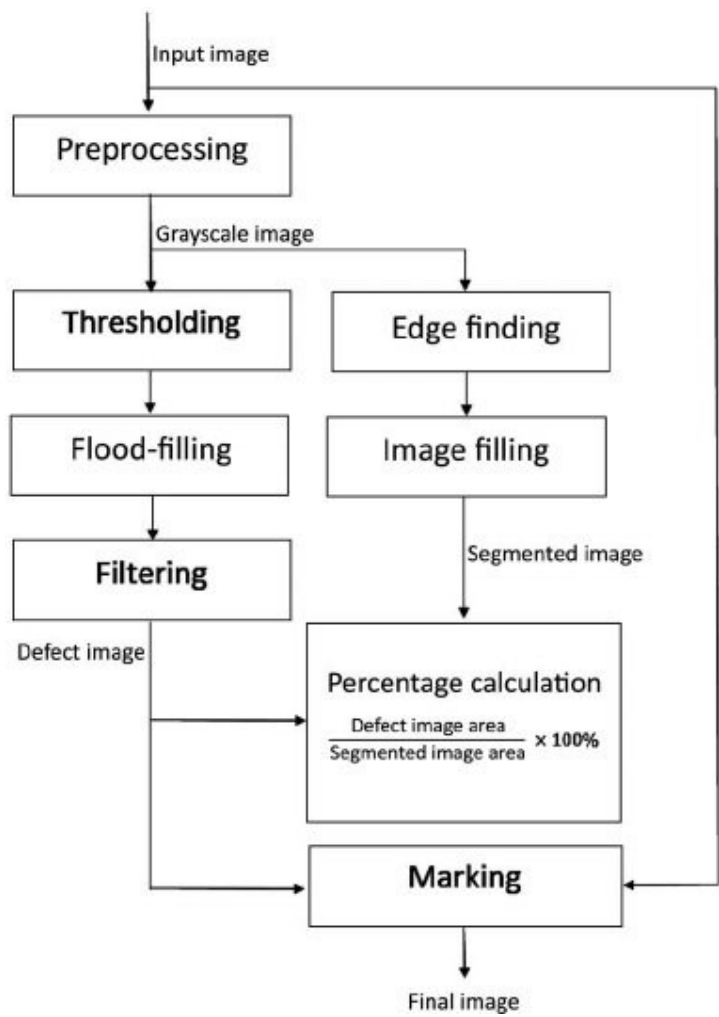
Four different percussion effects classified as snare drum “rim shots” were demonstrated by

the subject (Table TABLE I.). Two effects were realized as dorsal stops, differentiated by their airstream mechanisms. Two other rim shot sounds were produced as lingual ingressive consonants, or clicks.

The effect described as “rim shot K” was produced as a voiceless pulmonic egressive dorsal stop, similar to English /k/, but with an exaggerated, prolonged aspiration burst: [khh:]. Mean duration of the aspiration burst (interval over which aspiration noise exceeded 10% of maximum stop intensity), calculated across three tokens of this effect, was 576 ms, compared to mean VOT durations of 80 ms and 60 ms for voiceless (initial) dorsal stops in American (Lisker and Abramson, 1964) and Canadian English (Sundara, 2005), respectively.

A second effect produced at the same place of articulation was realized as an ejective stop [k'], illustrated in Fig. Fig.4—an4—an image sequence acquired over a 480 ms interval during the production of the second token. Dorsal closure (frame 80) occurs well before laryngeal lowering commences (frame 83). Upward movement of the closed glottis can be observed after the velum closes off the nasopharyngeal port, and glottal closure is maintained until after the dorsal constriction is released (frame 90).

FLOW DIAGRAM



REFERENCES

1. Dudgeon, D.E. and R.M. Mersereau, Multidimensional Digital Signal Processing. 1984, Englewood Cliffs, New Jersey: Prentice-Hall.
2. Castleman, K.R., Digital Image Processing. Second ed. 1996, Englewood Cliffs, New Jersey: Prentice-Hall.
3. Oppenheim, A.V., A.S. Willsky, and I.T. Young, Systems and Signals. 1983, Englewood Cliffs, New Jersey: Prentice-Hall.
4. Papoulis, A., Systems and Transforms with Applications in Optics. 1968, New York: McGraw-Hill.
5. Russ, J.C., The Image Processing Handbook. Second ed. 1995, Boca Raton, Florida: CRC Press.
6. Giardina, C.R. and E.R. Dougherty, Morphological Methods in Image and Signal Processing. 1988, Englewood Cliffs, New Jersey: Prentice-Hall. 321. ...Image Processing Fundamentals 110
7. Gonzalez, R.C. and R.E. Woods, Digital Image Processing. 1992, Reading, Massachusetts: Addison-Wesley. 716.
8. Goodman, J.W., Introduction to Fourier Optics. McGraw-Hill Physical and Quantum Electronics Series. 1968, New York: McGraw-Hill. 287.
9. Heijmans, H.J.A.M., Morphological Image Operators. Advances in Electronics and Electron Physics. 1994, Boston: Academic Press.
10. Hunt, R.W.G., The Reproduction of Colour in Photography, Printing & Television,. Fourth ed. 1987, Tolworth, England: Fountain Press.
11. Bae Y, Kuehn DP, Conway CA, Sutton BP. Real-time magnetic resonance imaging of velopharyngeal activities with simultaneous speech recordings. Cleft Palate Craniofac J 2011;48:695–707.
12. Maturo S, Silver A, Nimkin K, Sagar P, Ashland J, van der Kouwe AJ, Hartnick C. MRI with synchronized audio to evaluate velopharyngeal insufficiency. Cleft Palate Craniofac J 2012;49:761–763.
13. Drissi C, Mitrofanoff M, Talandier C, Falip C, Le Couls V, Adamsbaum C. Feasibility of dynamic MRI for evaluating velopharyngeal insufficiency in children. Eur Radiol 2011;21:1462–1469.
14. Tian W, Li Y, Yin H, Zhao SF, Li S, Wang Y, Shi B. Magnetic resonance imaging assessment of velopharyngeal motion in Chinese children after primary palatal repair. J Craniofac Surg 2010;21:578–587.
15. Kazan-Tannus JF, Levine D, McKenzie C, Lim KH, Cohen B, Farrar N, Busse RF, Mulliken JB. Real-time magnetic resonance imaging aids prenatal diagnosis of isolated cleft palate. J Ultrasound Med 2005;24:1533–1540.
16. Hagedorn C, Lammert A, Bassily M, Zu Y, Sinha U, Goldstein L, Narayanan SS. Characterizing post-glossectomy speech using realtime MRI. In Proceedings of the 10th International Seminar on Speech Production (ISSP), Cologne, Germany, May 2014.
17. Zu Y, Narayanan S, Kim YC, Nayak K, Bronson-Lowe C, Villegas B, Ouyoung M, Sinha U. Evaluation of swallow function post tongue cancer treatment using real-time MRI: a pilot study.

JAMA Otolaryngol Head Neck Surg 2013;139:1312-1319.

18. Adams SG, Weismer G, Kent RD. Speaking rate and speech movement velocity profiles. *J Speech Hear Res* 1993;36:41–54.

19. Tasko SM, McClean MD. Variations in articulatory movement with changes in speech task. *J Speech Lang Hear Res* 2004;47:85–100.

20. Scott AD, Boubertakh R, Birch MJ, Miquel ME. Towards clinical assessment of velopharyngeal closure using MRI: evaluation of realtime MRI sequences at 1.5 and 3 T. *Br J Radiol* 2012;85:e1083–e1092.

21. Narayanan S, Nayak K, Lee S, Sethy A, Byrd, D. An approach to realtime magnetic resonance imaging for speech production. *J Acoust Soc Am* 2004;115:1771–1776

22. Narayanan S, Toutios A, Ramanarayanan V, Lammert A, Kim J, Lee S, Nayak K, Kim YC, Zhu Y, Goldstein L, Byrd D, Bresch E, Ghosh P, Katsamanis A, Proctor M. Real-time magnetic resonance imaging and electromagnetic articulography database for speech production research. *J Acoust Soc Am* 2014;136:1307–1311.

23. Niebergall A, Zhang S, Kunay E, Keydana G, Job M, Uecker M, Frahm J. Real-time MRI of speaking at a resolution of 33 ms: Undersampled radial FLASH with nonlinear inverse reconstruction. *Magn Reson Med* 2013;69:477–485.

24. Burdumy M, Traser L, Richter B, Echternach M, Korvink JG, Hennig J, Zaitsev M. Acceleration of MRI of the vocal tract provides additional insight into articulator modifications. *J Magn Reson Imaging* 2015;42:925–935.

25. Freitas AC, Wylezinska M, Birch M, Petersen SE, Miquel ME. Real time speech MRI: a comparison of Cartesian and non-Cartesian sequences. In *Proceedings of ISMRM 23rd Scientific Sessions* (p. 655), Toronto, Ontario, Canada, May 30–June 5, 2015.

26. Iltis PW, Frahm J, Voit D, Joseph AA, Schoonderwaldt E, Altenmuller E. High-speed real-time magnetic resonance imaging of fast tongue movements in elite horn players. *Quant Imaging Med Surg* 2015;5:374–381.

27. Fu M, Zhao B, Carignan C, Shosted RK, Perry JL, Kuehn DP, Liang ZP, Sutton BP. High resolution dynamic speech imaging with joint low-rank and sparsity constraints. *Magn Reson Med* 2015;73:1820–1832.

28. Gupta AS, Liang ZP. Dynamic imaging by temporal modeling with principal component analysis. In *Proceedings of the 9th Annual Meeting of the International Society for Magnetic Resonance in Medicine* (p. 10), Glasgow, Scotland, UK, April 21–27, 2001.

29. Liang ZP. Spatiotemporal imaging with partially separable functions. In *Noninvasive Functional Source Imaging of the Brain and Heart and the International Conference on Functional Biomedical Imaging, 2007. NFSI-ICFBI 2007. Joint Meeting of the 6th International Symposium on* (pp. 181–182). New York: IEEE.

30. Zhao B, Haldar JP, Christodoulou AG, Liang ZP. Image reconstruction from highly undersampled-space data with joint partial separability and sparsity constraints. *IEEE Transact Med Imaging* 2012;31:1809–1820.