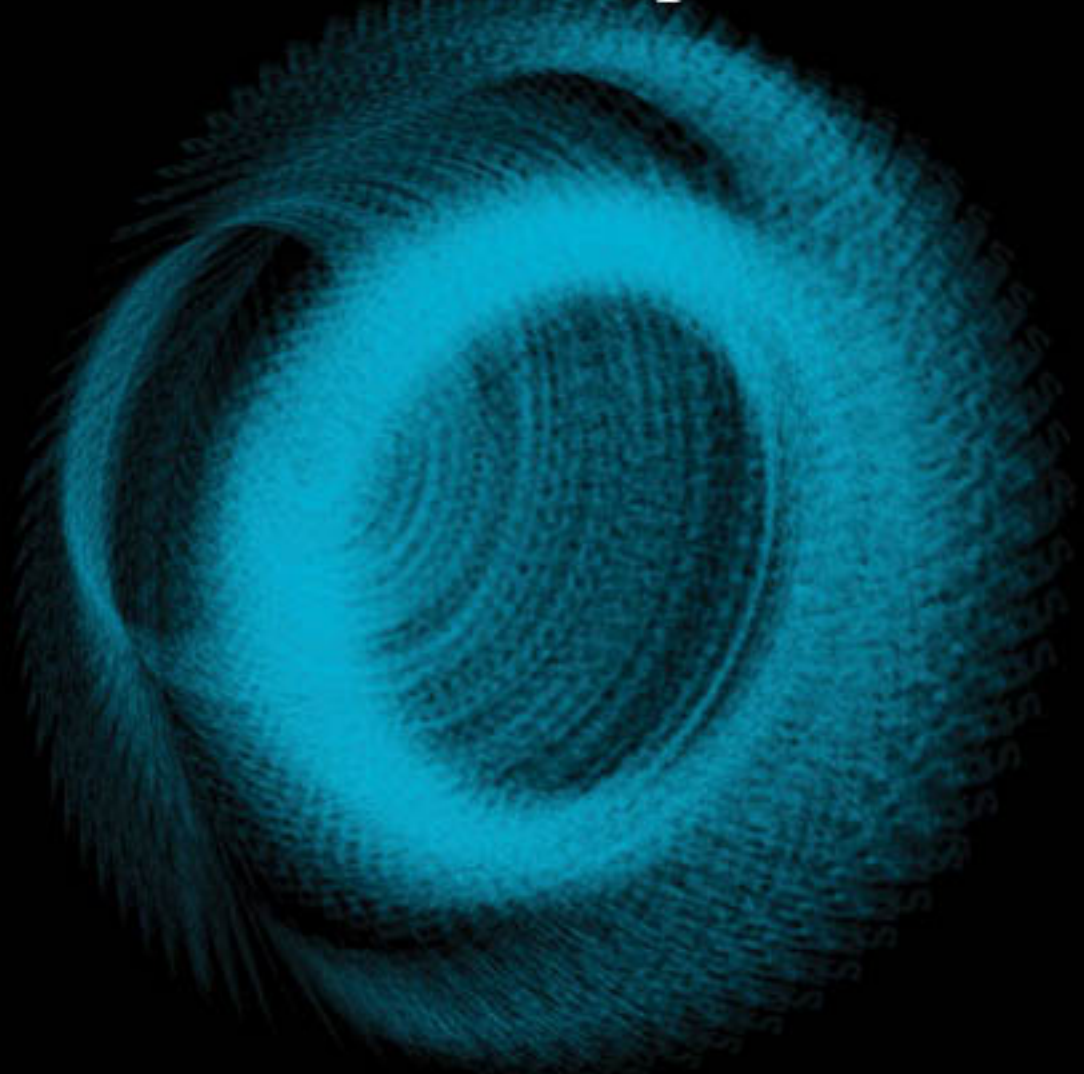
Important Concepts in Signal Processing, Image Processing and Data Compression



Dorsey Lopes

First Edition, 2012

ISBN 978-81-323-3604-4

Published by:
University Publications
4735/22 Prakashdeep Bldg,
Ansari Road, Darya Ganj,
Delhi - 110002
Email: info@wtbooks.com

Table of Contents

Chapter- 1

Audio Signal Processing

Audio signal processing, sometimes referred to as **audio processing**, is the intentional alteration of auditory signals, or sound. As audio signals may be electronically represented in either digital or analog format, signal processing may occur in either domain. Analog processors operate directly on the electrical signal, while digital processors operate mathematically on the digital representation of that signal.

History

Audio processing was necessary for early radio broadcasting -- as there were many problems with studio to transmitter links.

Analog signals

An analog representation is usually a continuous, non-discrete, electrical; a voltage level represents the air pressure waveform of the sound.

Digital signals

A digital representation expresses the pressure wave-form as a sequence of symbols, usually binary numbers. This permits signal processing using digital circuits such as microprocessors and computers. Although such a conversion can be prone to loss, most modern audio systems use this approach as the techniques of digital signal processing are much more powerful and efficient than analog domain signal processing.

Application areas

Processing methods and application areas include storage, level compression, data compression, transmission, enhancement (e.g., equalization, filtering, noise cancellation, echo or reverb removal or addition, etc.)

Audio Broadcasting

Audio broadcasting (be it for television or audio broadcasting) is perhaps the biggest market segment (and user area) for audio processing products—globally.

Traditionally the most important audio processing (in audio broadcasting) takes place just before the transmitter. Studio audio processing is limited in the modern era due to digital audio systems (mixers, routers) being pervasive in the studio.

In audio broadcasting, the audio processor must

- prevent overmodulation, and minimize it when it occurs
- compensate for non-linear transmitters, more common with medium wave and shortwave broadcasting
- adjust overall loudness to desired level
- correct errors in audio levels

Chapter- 2

Digital Image Processing

Digital image processing is the use of computer algorithms to perform image processing on digital images. As a subcategory or field of digital signal processing, digital image processing has many advantages over analog image processing. It allows a much wider range of algorithms to be applied to the input data and can avoid problems such as the build-up of noise and signal distortion during processing. Since images are defined over two dimensions (perhaps more) digital image processing may be modeled in the form of Multidimensional Systems.

History

Many of the techniques of digital image processing, or digital picture processing as it often was called, were developed in the 1960s at the Jet Propulsion Laboratory, Massachusetts Institute of Technology, Bell Laboratories, University of Maryland, and a few other research facilities, with application to satellite imagery, wire-photo standards conversion, medical imaging, videophone, character recognition, and photograph enhancement. The cost of processing was fairly high, however, with the computing equipment of that era. That changed in the 1970s, when digital image processing proliferated as cheaper computers and dedicated hardware became available. Images then could be processed in real time, for some dedicated problems such as television standards conversion. As general-purpose computers became faster, they started to take over the role of dedicated hardware for all but the most specialized and computer-intensive operations.

With the fast computers and signal processors available in the 2000s, digital image processing has become the most common form of image processing and generally, is used because it is not only the most versatile method, but also the cheapest.

Digital image processing technology for medical applications was inducted into the Space Foundation Space Technology Hall of Fame in 1994.

Tasks

Digital image processing allows the use of much more complex algorithms for image processing, and hence, can offer both more sophisticated performance at simple tasks, and the implementation of methods which would be impossible by analog means.

In particular, digital image processing is the only practical technology for:

- Classification
- Feature extraction
- Pattern recognition
- Projection
- Multi-scale signal analysis

Some techniques which are used in digital image processing include:

- Pixelization
- Linear filtering
- Principal components analysis
- Independent component analysis
- Hidden Markov models
- Anisotropic diffusion
- Partial differential equations
- Self-organizing maps
- Neural networks
- Wavelets

Applications

Digital camera images

Digital cameras generally include dedicated digital image processing chips to convert the raw data from the image sensor into a color-corrected image in a standard image file format. Images from digital cameras often receive further processing to improve their quality, a distinct advantage that digital cameras have over film cameras. The digital image processing typically is executed by special software programs that can manipulate the images in many ways.

Many digital cameras also enable viewing of histograms of images, as an aid for the photographer to understand the rendered brightness range of each shot more readily.

Film

Westworld (1973) was the first feature film to use digital image processing to pixellate photography to simulate an android's point of view.

Intelligent Transportation Systems

Digital image processing has a wide applications in intelligent transportation systems, such as Automatic number plate recognition and Traffic sign recognition.

Chapter- 3

Computer Vision

Computer vision is the science and technology of machines that see, where *see* in this case means that the machine is able to extract information from an image that is necessary to solve some task. As a scientific discipline, computer vision is concerned with the theory behind artificial systems that extract information from images. The image data can take many forms, such as video sequences, views from multiple cameras, or multi-dimensional data from a medical scanner.

As a technological discipline, computer vision seeks to apply its theories and models to the construction of computer vision systems. Examples of applications of computer vision include systems for:

- Controlling processes (e.g., an industrial robot or an autonomous vehicle).
- Detecting events (e.g., for visual surveillance or people counting).
- Organizing information (e.g., for indexing databases of images and image sequences).
- Modeling objects or environments (e.g., industrial inspection, medical image analysis or topographical modeling).
- Interaction (e.g., as the input to a device for computer-human interaction).

Computer vision is closely related to the study of biological vision. The field of biological vision studies and models the physiological processes behind visual perception in humans and other animals. Computer vision, on the other hand, studies and describes the processes implemented in software and hardware behind artificial vision systems. Interdisciplinary exchange between biological and computer vision has proven fruitful for both fields.

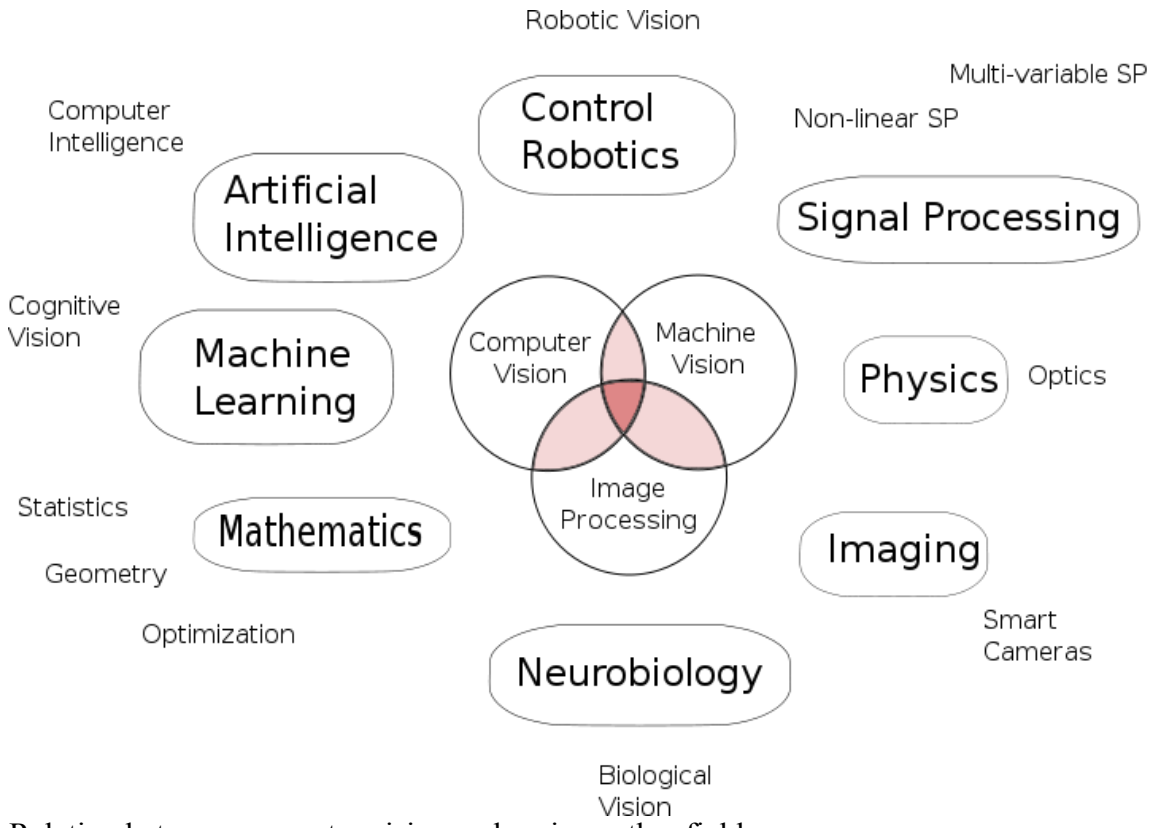
Computer vision is, in some ways, the inverse of computer graphics. While computer graphics produces image data from 3D models, computer vision often produces 3D models from image data. There is also a trend towards a combination of the two disciplines, e.g., as explored in augmented reality.

Sub-domains of computer vision include scene reconstruction, event detection, video tracking, object recognition, learning, indexing, motion estimation, and image restoration.

State of the art

Computer vision is a diverse and relatively new field of study. In the early days of computing, it was difficult to process even moderately large sets of image data. It was not until the late 1970s that a more focused study of the field emerged. Computer vision covers a wide range of topics which are often related to other disciplines, and consequently there is no standard formulation of "the computer vision problem". Moreover, there is no standard formulation of how computer vision problems should be solved. Instead, there exists an abundance of methods for solving various well-defined computer vision tasks, where the methods often are very task specific and seldom can be generalised over a wide range of applications. Many of the methods and applications are still in the state of basic research, but more and more methods have found their way into commercial products, where they often constitute a part of a larger system which can solve complex tasks (e.g., in the area of medical images, or quality control and measurements in industrial processes). In most practical computer vision applications, the computers are pre-programmed to solve a particular task, but methods based on learning are now becoming increasingly common.

Related fields



Relation between computer vision and various other fields

Much of artificial intelligence deals with autonomous planning or deliberation for robotical systems to navigate through an environment. A detailed understanding of these

environments is required to navigate through them. Information about the environment could be provided by a computer vision system, acting as a vision sensor and providing high-level information about the environment and the robot. Artificial intelligence and computer vision share other topics such as pattern recognition and learning techniques. Consequently, computer vision is sometimes seen as a part of the artificial intelligence field or the computer science field in general.

Physics is another field that is closely related to computer vision. Computer vision systems rely on image sensors which detect electromagnetic radiation which is typically in the form of either visible or infra-red light. The sensors are designed using solid-state physics. The process by which light propagates and reflects off surfaces is explained using optics. Sophisticated image sensors even require quantum mechanics to provide a complete understanding of the image formation process. Also, various measurement problems in physics can be addressed using computer vision, for example motion in fluids.

A third field which plays an important role is neurobiology, specifically the study of the biological vision system. Over the last century, there has been an extensive study of eyes, neurons, and the brain structures devoted to processing of visual stimuli in both humans and various animals. This has led to a coarse, yet complicated, description of how "real" vision systems operate in order to solve certain vision related tasks. These results have led to a subfield within computer vision where artificial systems are designed to mimic the processing and behavior of biological systems, at different levels of complexity. Also, some of the learning-based methods developed within computer vision have their background in biology.

Yet another field related to computer vision is signal processing. Many methods for processing of one-variable signals, typically temporal signals, can be extended in a natural way to processing of two-variable signals or multi-variable signals in computer vision. However, because of the specific nature of images there are many methods developed within computer vision which have no counterpart in the processing of one-variable signals. A distinct character of these methods is the fact that they are non-linear which, together with the multi-dimensionality of the signal, defines a subfield in signal processing as a part of computer vision.

Beside the above mentioned views on computer vision, many of the related research topics can also be studied from a purely mathematical point of view. For example, many methods in computer vision are based on statistics, optimization or geometry. Finally, a significant part of the field is devoted to the implementation aspect of computer vision; how existing methods can be realised in various combinations of software and hardware, or how these methods can be modified in order to gain processing speed without losing too much performance.

The fields most closely related to computer vision are image processing, image analysis and machine vision. There is a significant overlap in the range of techniques and applications that these cover. This implies that the basic techniques that are used and

developed in these fields are more or less identical, something which can be interpreted as there is only one field with different names. On the other hand, it appears to be necessary for research groups, scientific journals, conferences and companies to present or market themselves as belonging specifically to one of these fields and, hence, various characterizations which distinguish each of the fields from the others have been presented.

The following characterizations appear relevant but should not be taken as universally accepted:

- Image processing and image analysis tend to focus on 2D images, how to transform one image to another, e.g., by pixel-wise operations such as contrast enhancement, local operations such as edge extraction or noise removal, or geometrical transformations such as rotating the image. This characterisation implies that image processing/analysis neither require assumptions nor produce interpretations about the image content.
- Computer vision tends to focus on the 3D scene projected onto one or several images, e.g., how to reconstruct structure or other information about the 3D scene from one or several images. Computer vision often relies on more or less complex assumptions about the scene depicted in an image.
- Machine vision tends to focus on applications, mainly in manufacturing, e.g., vision based autonomous robots and systems for vision based inspection or measurement. This implies that image sensor technologies and control theory often are integrated with the processing of image data to control a robot and that real-time processing is emphasised by means of efficient implementations in hardware and software. It also implies that the external conditions such as lighting can be and are often more controlled in machine vision than they are in general computer vision, which can enable the use of different algorithms.
- There is also a field called imaging which primarily focus on the process of producing images, but sometimes also deals with processing and analysis of images. For example, medical imaging contains lots of work on the analysis of image data in medical applications.
- Finally, pattern recognition is a field which uses various methods to extract information from signals in general, mainly based on statistical approaches. A significant part of this field is devoted to applying these methods to image data.

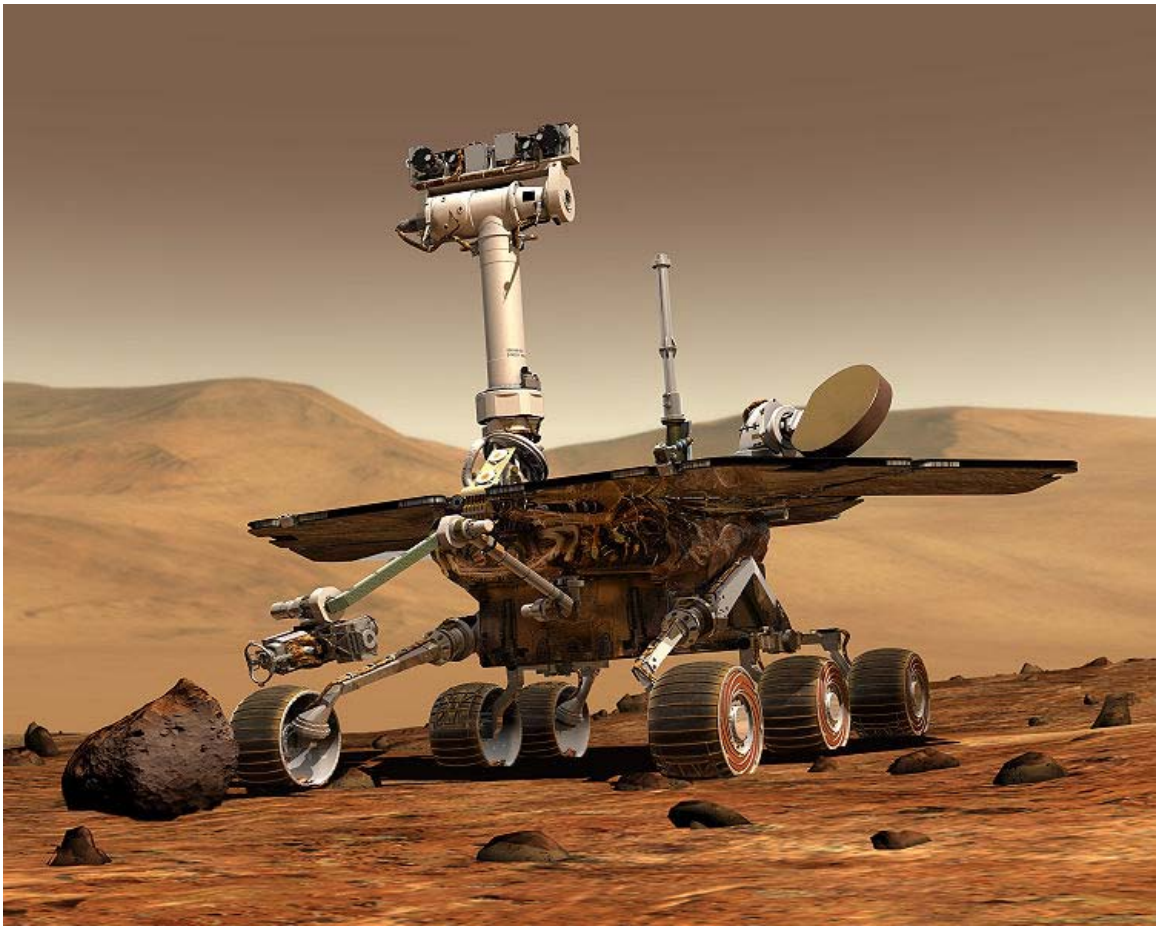
Applications for computer vision

One of the most prominent application fields is medical computer vision or medical image processing. This area is characterized by the extraction of information from image data for the purpose of making a medical diagnosis of a patient. Generally, image data is in the form of microscopy images, X-ray images, angiography images, ultrasonic images, and tomography images. An example of information which can be extracted from such image data is detection of tumours, arteriosclerosis or other malign changes. It can also be measurements of organ dimensions, blood flow, etc. This application area also

supports medical research by providing new information, e.g., about the structure of the brain, or about the quality of medical treatments.

A second application area in computer vision is in industry, sometimes called machine vision, where information is extracted for the purpose of supporting a manufacturing process. One example is quality control where details or final products are being automatically inspected in order to find defects. Another example is measurement of position and orientation of details to be picked up by a robot arm. Machine vision is also heavily used in agricultural process to remove undesirable food stuff from bulk material, a process called optical sorting.

Military applications are probably one of the largest areas for computer vision. The obvious examples are detection of enemy soldiers or vehicles and missile guidance. More advanced systems for missile guidance send the missile to an area rather than a specific target, and target selection is made when the missile reaches the area based on locally acquired image data. Modern military concepts, such as "battlefield awareness", imply that various sensors, including image sensors, provide a rich set of information about a combat scene which can be used to support strategic decisions. In this case, automatic processing of the data is used to reduce complexity and to fuse information from multiple sensors to increase reliability.



Artist's Concept of Rover on Mars, an example of an unmanned land-based vehicle. Notice the stereo cameras mounted on top of the Rover.

One of the newer application areas is autonomous vehicles, which include submersibles, land-based vehicles (small robots with wheels, cars or trucks), aerial vehicles, and unmanned aerial vehicles (UAV). The level of autonomy ranges from fully autonomous (unmanned) vehicles to vehicles where computer vision based systems support a driver or a pilot in various situations. Fully autonomous vehicles typically use computer vision for navigation, i.e. for knowing where it is, or for producing a map of its environment (SLAM) and for detecting obstacles. It can also be used for detecting certain task specific events, e. g., a UAV looking for forest fires. Examples of supporting systems are obstacle warning systems in cars, and systems for autonomous landing of aircraft. Several car manufacturers have demonstrated systems for autonomous driving of cars, but this technology has still not reached a level where it can be put on the market. There are ample examples of military autonomous vehicles ranging from advanced missiles, to UAVs for recon missions or missile guidance. Space exploration is already being made with autonomous vehicles using computer vision, e. g., NASA's Mars Exploration Rover and ESA's ExoMars Rover.

Other application areas include:

- Support of visual effects creation for cinema and broadcast, e.g., camera tracking (matchmoving).
- Surveillance.

Typical tasks of computer vision

Each of the application areas described above employ a range of computer vision tasks; more or less well-defined measurement problems or processing problems, which can be solved using a variety of methods. Some examples of typical computer vision tasks are presented below.

Recognition

The classical problem in computer vision, image processing, and machine vision is that of determining whether or not the image data contains some specific object, feature, or activity. This task can normally be solved robustly and without effort by a human, but is still not satisfactorily solved in computer vision for the general case: arbitrary objects in arbitrary situations. The existing methods for dealing with this problem can at best solve it only for specific objects, such as simple geometric objects (e.g., polyhedra), human faces, printed or hand-written characters, or vehicles, and in specific situations, typically described in terms of well-defined illumination, background, and pose of the object relative to the camera.

Different varieties of the recognition problem are described in the literature:

- **Object recognition**: one or several pre-specified or learned objects or object classes can be recognized, usually together with their 2D positions in the image or 3D poses in the scene.
- **Identification**: An individual instance of an object is recognized. Examples: identification of a specific person's face or fingerprint, or identification of a specific vehicle.
- **Detection**: the image data is scanned for a specific condition. Examples: detection of possible abnormal cells or tissues in medical images or detection of a vehicle in an automatic road toll system. Detection based on relatively simple and fast computations is sometimes used for finding smaller regions of interesting image data which can be further analysed by more computationally demanding techniques to produce a correct interpretation.

Several specialized tasks based on recognition exist, such as:

- **Content-based image retrieval**: finding all images in a larger set of images which have a specific content. The content can be specified in different ways, for example in terms of similarity relative a target image (give me all images similar to image X), or in terms of high-level search criteria given as text input (give me all images which contains many houses, are taken during winter, and have no cars in them).
- **Pose estimation**: estimating the position or orientation of a specific object relative to the camera. An example application for this technique would be assisting a robot arm in retrieving objects from a conveyor belt in an assembly line situation.
- **Optical character recognition** (OCR): identifying characters in images of printed or handwritten text, usually with a view to encoding the text in a format more amenable to editing or indexing (e.g. ASCII).

Motion analysis

Several tasks relate to motion estimation where an image sequence is processed to produce an estimate of the velocity either at each points in the image or in the 3D scene, or even of the camera that produces the images . Examples of such tasks are:

- **Egomotion**: determining the 3D rigid motion (rotation and translation) of the camera from an image sequence produced by the camera.
- **Tracking**: following the movements of a (usually) smaller set of interest points or objects (e.g., vehicles or humans) in the image sequence.
- **Optical flow**: to determine, for each point in the image, how that point is moving relative to the image plane, i.e., its apparent motion. This motion is a result both of how the corresponding 3D point is moving in the scene and how the camera is moving relative to the scene.

Scene reconstruction

Given one or (typically) more images of a scene, or a video, scene reconstruction aims at computing a 3D model of the scene. In the simplest case the model can be a set of 3D points. More sophisticated methods produce a complete 3D surface model.

Image restoration

The aim of image restoration is the removal of noise (sensor noise, motion blur, etc.) from images. The simplest possible approach for noise removal is various types of filters such as low-pass filters or median filters. More sophisticated methods assume a model of how the local image structures look like, a model which distinguishes them from the noise. By first analysing the image data in terms of the local image structures, such as lines or edges, and then controlling the filtering based on local information from the analysis step, a better level of noise removal is usually obtained compared to the simpler approaches. An example in this field is the inpainting.

Computer vision systems

The organization of a computer vision system is highly application dependent. Some systems are stand-alone applications which solve a specific measurement or detection problem, while others constitute a sub-system of a larger design which, for example, also contains sub-systems for control of mechanical actuators, planning, information databases, man-machine interfaces, etc. The specific implementation of a computer vision system also depends on if its functionality is pre-specified or if some part of it can be learned or modified during operation. There are, however, typical functions which are found in many computer vision systems.

- **Image acquisition**: A digital image is produced by one or several image sensors, which, besides various types of light-sensitive cameras, include range sensors, tomography devices, radar, ultra-sonic cameras, etc. Depending on the type of sensor, the resulting image data is an ordinary 2D image, a 3D volume, or an image sequence. The pixel values typically correspond to light intensity in one or several spectral bands (gray images or colour images), but can also be related to various physical measures, such as depth, absorption or reflectance of sonic or electromagnetic waves, or nuclear magnetic resonance.
- **Pre-processing**: Before a computer vision method can be applied to image data in order to extract some specific piece of information, it is usually necessary to process the data in order to assure that it satisfies certain assumptions implied by the method. Examples are
    - Re-sampling in order to assure that the image coordinate system is correct.
    - Noise reduction in order to assure that sensor noise does not introduce false information.
    - Contrast enhancement to assure that relevant information can be detected.
    - Scale-space representation to enhance image structures at locally appropriate scales.
- **Feature extraction**: Image features at various levels of complexity are extracted from the image data. Typical examples of such features are

- Lines, edges and ridges.
- Localized interest points such as corners, blobs or points.

More complex features may be related to texture, shape or motion.

- **Detection/segmentation**: At some point in the processing a decision is made about which image points or regions of the image are relevant for further processing. Examples are
  - Selection of a specific set of interest points
  - Segmentation of one or multiple image regions which contain a specific object of interest.
- **High-level processing**: At this step the input is typically a small set of data, for example a set of points or an image region which is assumed to contain a specific object. The remaining processing deals with, for example:
  - Verification that the data satisfy model-based and application specific assumptions.
  - Estimation of application specific parameters, such as object pose or object size.
  - Image recognition: classifying a detected object into different categories.
  - Image registration: comparing and combining two different views of the same object.

Chapter- 4

Noise Reduction

Noise reduction is the process of removing noise from a signal. Noise reduction techniques are conceptually very similar regardless of the signal being processed, however a priori knowledge of the characteristics of an expected signal can mean the implementations of these techniques vary greatly depending on the type of signal.

All recording devices, both analogue or digital, have traits which make them susceptible to noise. Noise can be random or white noise with no coherence, or coherent noise introduced by the device's mechanism or processing algorithms.

In electronic recording devices, a major form of noise is *hiss* caused by random electrons that, heavily influenced by heat, stray from their designated path. These stray electrons influence the voltage of the output signal and thus create detectable noise.

In the case of photographic film and magnetic tape, noise (both visible and audible) is introduced due to the grain structure of the medium. In photographic film, the size of the grains in the film determines the film's sensitivity, more sensitive film having larger sized grains. In magnetic tape, the larger the grains of the magnetic particles (usually ferric oxide or magnetite), the more prone the medium is to noise.

To compensate for this, larger areas of film or magnetic tape may be used to lower the noise to an acceptable level.

In audio

When using analog tape recording technology, they may exhibit a type of noise known as tape hiss. This is related to the particle size and texture used in the magnetic emulsion that is sprayed on the recording media, and also to the relative tape velocity across the tape heads.

Four types of noise reduction exist: single-ended pre-recording, single-ended hiss reduction, single-ended surface noise reduction, and codec or dual-ended systems. Single-ended pre-recording systems (such as Dolby HX Pro) work to affect the recording medium at the time of recording. Single-ended hiss reduction systems (such as DNR) work to reduce noise as it occurs, including both before and after the recording process as

well as for live broadcast applications. Single-ended surface noise reduction (such as CEDAR and the earlier SAE 5000A and Burwen TNE 7000) is applied to the playback of phonograph records to attenuate the sound of scratches, pops, and surface non-linearities. Dual-ended systems (such as Dolby NR and dbx Type I and II) have a pre-emphasis process applied during recording and then a de-emphasis process applied at playback.

Dolby and dbx noise reduction system

While there are dozens of different kinds of noise reduction, the first widely used audio noise reduction technique was developed by Ray Dolby in 1966. Intended for professional use, Dolby Type A was an encode/decode system in which the amplitude of frequencies in four bands was increased during recording (encoding), then decreased proportionately during playback (decoding). The Dolby B system (developed in conjunction with Henry Kloss) was a single band system designed for consumer products. In particular, when recording quiet parts of an audio signal, the frequencies above 1 kHz would be boosted. This had the effect of increasing the signal to noise ratio on tape up to 10dB depending on the initial signal volume. When it was played back, the decoder reversed the process, in effect reducing the noise level by up to 10dB. The Dolby B system, while not as effective as Dolby A, had the advantage of remaining listenable on playback systems without a decoder.

Dbx was the competing analog noise reduction system developed by dbx laboratories. It used a root-mean-squared (RMS) encode/decode algorithm with the noise-prone high frequencies boosted, and the entire signal fed through a 2:1 compander. Dbx operated across the entire audible bandwidth and unlike Dolby B was unusable as an open ended system. However it could achieve up to 30 dB of noise reduction. Since Analog video recordings use frequency modulation for the luminance part (composite video signal in direct colour systems), which keeps the tape at saturation level, audio style noise reduction is unnecessary.

Dynamic Noise Reduction

Dynamic Noise Reduction (DNR) is an audio noise reduction system, introduced by National Semiconductor to reduce noise levels on long-distance telephony. First sold in 1981, DNR is frequently confused with the far more common Dolby noise reduction system. However, unlike Dolby and dbx Type I & Type II noise reduction systems, DNR is a playback-only signal processing system that does not require the source material to first be encoded, and it can be used together with other forms of noise reduction. It was a development of the unpatented Philips Dynamic Noise Limiter (DNL) system, introduced in 1971, with the circuitry on a single chip.

Because DNR is non-complementary, meaning it does not require encoded source material, it can be used to remove background noise from any audio signal, including magnetic tape recordings and FM radio broadcasts, reducing noise by as much as 10 dB. It can be used in conjunction with other noise reduction systems, provided that they are

used prior to applying DNR to prevent DNR from causing the other noise reduction system to mistrack.

One of DNR's first widespread applications was in the GM Delco Bose car stereo systems in U.S. GM cars (later added to Delco-manufactured car stereos in GM vehicles as well), introduced in 1984. It was also used in factory car stereos in Jeep vehicles in the 1980s, such as the Cherokee XJ. Today, DNR, DNL, and similar systems are most commonly encountered as a noise reduction system in microphone systems.

Other approaches

A second class of algorithms work in the time-frequency domain using some linear or non-linear filters that have local characteristics and are often called time-frequency filters. Noise can therefore be also removed by use of spectral editing tools, which work in this time-frequency domain, allowing local modifications without affecting nearby signal energy. This can be done manually by using the mouse with a pen that has a defined time-frequency shape. This is done much like in a paint program drawing pictures. Another way is to define a dynamic threshold for filtering noise, that is derived from the local signal, again with respect to a local time-frequency region. Everything below the threshold will be filtered, everything above the threshold, like partials of a voice or "wanted noise", will be untouched. The region is typically defined by the location of the signal Instantaneous Frequency, as most of the signal energy to be preserved is concentrated about it.

Modern digital sound (and picture) recordings no longer need to worry about tape hiss so analog style noise reduction systems are not necessary. However, an interesting twist is that dither systems actually add noise to a signal to improve its quality.

In images

Images taken with both digital cameras and conventional film cameras will pick up noise from a variety of sources. Many further uses of these images require that the noise will be (partially) removed - for aesthetic purposes as in artistic work or marketing, or for practical purposes such as computer vision.

Types

In salt and pepper noise (sparse light and dark disturbances), pixels in the image are very different in color or intensity from their surrounding pixels; the defining characteristic is that the value of a noisy pixel bears no relation to the color of surrounding pixels. Generally this type of noise will only affect a small number of image pixels. When viewed, the image contains dark and white dots, hence the term salt and pepper noise. Typical sources include flecks of dust inside the camera and overheated or faulty CCD elements.

In Gaussian noise, each pixel in the image will be changed from its original value by a (usually) small amount. A histogram, a plot of the amount of distortion of a pixel value against the frequency with which it occurs, shows a normal distribution of noise. While other distributions are possible, the Gaussian (normal) distribution is usually a good model, due to the central limit theorem that says that the sum of different noises tends to approach a Gaussian distribution.

In either case, the noises at different pixels can be either correlated or uncorrelated; in many cases, noise values at different pixels are modeled as being independent and identically distributed, and hence uncorrelated.

Removal

Tradeoffs

In selecting a noise reduction algorithm, one must weigh several factors:

- the available computer power and time available: a digital camera must apply noise reduction in a fraction of a second using a tiny onboard CPU, while a desktop computer has much more power and time
- whether sacrificing some real detail is acceptable if it allows more noise to be removed (how aggressively to decide whether variations in the image are noise or not)
- the characteristics of the noise and the detail in the image, to better make those decisions

Chroma and luminance noise separation

In real-world photographs, the highest spatial-frequency detail consists mostly of variations in brightness ("luminance detail") rather than variations in hue ("chroma detail"). Since any noise reduction algorithm should attempt to remove noise without sacrificing real detail from the scene photographed, one risks a greater loss of detail from luminance noise reduction than chroma noise reduction simply because most scenes have little high frequency chroma detail to begin with. In addition, most people find chroma noise in images more objectionable than luminance noise; the colored blobs are considered "digital-looking" and unnatural, compared to the grainy appearance of luminance noise that some compare to film grain. For these two reasons, most photographic noise reduction algorithms split the image detail into chroma and luminance components and apply more noise reduction to the former; the in-camera noise reduction algorithm used by Nikon's DSLR's, in particular, is known for this.

Most dedicated noise-reduction computer software allows the user to control chroma and luminance noise reduction separately.

Linear smoothing filters

One method to remove noise is by convolving the original image with a mask that represents a low-pass filter or smoothing operation. For example, the Gaussian mask comprises elements determined by a Gaussian function. This convolution brings the value of each pixel into closer harmony with the values of its neighbors. In general, a smoothing filter sets each pixel to the average value, or a weighted average, of itself and its nearby neighbors; the Gaussian filter is just one possible set of weights.

Smoothing filters tend to blur an image, because pixel intensity values that are significantly higher or lower than the surrounding neighborhood would "smear" across the area. Because of this blurring, linear filters are seldom used in practice for noise reduction; they are, however, often used as the basis for nonlinear noise reduction filters.

Anisotropic diffusion

Another method for removing noise is to evolve the image under a smoothing partial differential equation similar to the heat equation which is called anisotropic diffusion. With a spatially constant diffusion coefficient, this is equivalent to the heat equation or linear Gaussian filtering, but with a diffusion coefficient designed to detect edges, the noise can be removed without blurring the edges of the image.

Nonlinear filters

A median filter is an example of a non-linear filter and, if properly designed, is very good at preserving image detail. To run a median filter:

1. consider each pixel in the image
2. sort the neighbouring pixels into order based upon their intensities
3. replace the original value of the pixel with the median value from the list

A median filter is a rank-selection (RS) filter, a particularly harsh member of the family of rank-conditioned rank-selection (RCRS) filters; a much milder member of that family, for example one that selects the closest of the neighboring values when a pixel's value is external in its neighborhood, and leaves it unchanged otherwise, is sometimes preferred, especially in photographic applications.

Median and other RCRS filters are good at removing salt and pepper noise from an image, and also cause relatively little blurring of edges, and hence are often used in computer vision applications.

Software programs

Most general purpose image and photo editing software will have one or more noise reduction functions (median, blur, despeckle, etc.). Special purpose noise reduction software programs include Neat Image, Grain Surgery, Noise Ninja, DenoiseMyImage,

GREYCstoration (now G'MIC), and pnmnlfilt (nonlinear filter) found in the open source Netpbm tools. General purpose image and photo editing software including noise reduction functions include Adobe Photoshop, GIMP, PhotoImpact, Paint Shop Pro, and Helicon Filter.

Chapter- 5

Edge Detection

Edge detection is a fundamental tool in image processing and computer vision, particularly in the areas of feature detection and feature extraction, which aim at identifying points in a digital image at which the image brightness changes sharply or, more formally, has discontinuities.

Motivations

Canny edge detection applied to a photograph

The purpose of detecting sharp changes in image brightness is to capture important events and changes in properties of the world. It can be shown that under rather general assumptions for an image formation model, discontinuities in image brightness are likely to correspond to:

- discontinuities in depth,
- discontinuities in surface orientation,
- changes in material properties and
- variations in scene illumination.

In the ideal case, the result of applying an edge detector to an image may lead to a set of connected curves that indicate the boundaries of objects, the boundaries of surface markings as well as curves that correspond to discontinuities in surface orientation. Thus, applying an edge detection algorithm to an image may significantly reduce the amount of

data to be processed and may therefore filter out information that may be regarded as less relevant, while preserving the important structural properties of an image. If the edge detection step is successful, the subsequent task of interpreting the information contents in the original image may therefore be substantially simplified. However, it is not always possible to obtain such ideal edges from real life images of moderate complexity. Edges extracted from non-trivial images are often hampered by *fragmentation*, meaning that the edge curves are not connected, missing edge segments as well as *false edges* not corresponding to interesting phenomena in the image – thus complicating the subsequent task of interpreting the image data.

Edge detection is one of the fundamental steps in image processing, image analysis, image pattern recognition, and computer vision techniques. During recent years, however, substantial (and successful) research has also been made on computer vision methods that do not explicitly rely on edge detection as a pre-processing step.

Edge properties

The edges extracted from a two-dimensional image of a three-dimensional scene can be classified as either viewpoint dependent or viewpoint independent. A *viewpoint independent edge* typically reflects inherent properties of the three-dimensional objects, such as surface markings and surface shape. A *viewpoint dependent edge* may change as the viewpoint changes, and typically reflects the geometry of the scene, such as objects occluding one another.

A typical edge might for instance be the border between a block of red color and a block of yellow. In contrast a **line** (as can be extracted by a ridge detector) can be a small number of pixels of a different color on an otherwise unchanging background. For a line, there may therefore usually be one edge on each side of the line.

A simple edge model

Although certain literature has considered the detection of ideal step edges, the edges obtained from natural images are usually not at all ideal step edges. Instead they are normally affected by one or several of the following effects:

- focal blur caused by a finite depth-of-field and finite point spread function.
- penumbral blur caused by shadows created by light sources of non-zero radius.
- shading at a smooth object

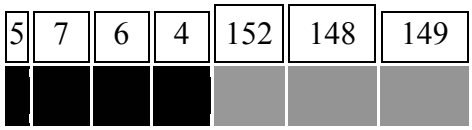
A number of researchers have used a Gaussian smoothed step edge (an error function) as the simplest extension of the ideal step edge model for modeling the effects of edge blur in practical applications. Thus, a one-dimensional image f which has exactly one edge placed at $x = 0$ may be modeled as:

$$f(x) = \frac{I_r - I_l}{2}\left(\operatorname{erf}\left(\frac{x}{\sqrt{2}\sigma}\right) + 1\right) + I_l.$$

At the left side of the edge, the intensity is $I_l = \lim_{x \to -\infty} f(x)$, and right of the edge it is $I_r = \lim_{x \to \infty} f(x)$. The scale parameter σ is called the blur scale of the edge.

Why edge detection is a non-trivial task

To illustrate why edge detection is not a trivial task, let us consider the problem of detecting edges in the following one-dimensional signal. Here, we may intuitively say that there should be an edge between the 4th and 5th pixels.



If the intensity difference were smaller between the 4th and the 5th pixels and if the intensity differences between the adjacent neighboring pixels were higher, it would not be as easy to say that there should be an edge in the corresponding region. Moreover, one could argue that this case is one in which there are several edges.

Hence, to firmly state a specific threshold on how large the intensity change between two neighbouring pixels must be for us to say that there should be an edge between these pixels is not always simple. Indeed, this is one of the reasons why edge detection may be a non-trivial problem unless the objects in the scene are particularly simple and the illumination conditions can be well controlled (see for example, the edges extracted from the image with the girl above).

Approaches to edge detection

There are many methods for edge detection, but most of them can be grouped into two categories, search-based and zero-crossing based. The search-based methods detect edges by first computing a measure of edge strength, usually a first-order derivative expression such as the gradient magnitude, and then searching for local directional maxima of the gradient magnitude using a computed estimate of the local orientation of the edge, usually the gradient direction. The zero-crossing based methods search for zero crossings in a second-order derivative expression computed from the image in order to find edges, usually the zero-crossings of the Laplacian or the zero-crossings of a non-linear

differential expression. As a pre-processing step to edge detection, a smoothing stage, typically Gaussian smoothing, is almost always applied.

The edge detection methods that have been published mainly differ in the types of smoothing filters that are applied and the way the measures of edge strength are computed. As many edge detection methods rely on the computation of image gradients, they also differ in the types of filters used for computing gradient estimates in the x- and y-directions.

Canny edge detection

John Canny considered the mathematical problem of deriving an optimal smoothing filter given the criteria of detection, localization and minimizing multiple responses to a single edge. He showed that the optimal filter given these assumptions is a sum of four exponential terms. He also showed that this filter can be well approximated by first-order derivatives of Gaussians. Canny also introduced the notion of non-maximum suppression, which means that given the presmoothing filters, edge points are defined as points where the gradient magnitude assumes a local maximum in the gradient direction. Looking for the zero crossing of the 2nd derivative along the gradient direction was first proposed by Haralick . It took less than two decades to find a modern geometric variational meaning for that operator that links it to the Marr-Hildreth (zero crossing of the Laplacian) edge detector. That observation was presented by Ron Kimmel and Alfred Bruckstein.

Although his work was done in the early days of computer vision, the Canny edge detector (including its variations) is still a state-of-the-art edge detector. Unless the preconditions are particularly suitable, it is hard to find an edge detector that performs significantly better than the Canny edge detector.

The Canny-Deriche detector was derived from similar mathematical criteria as the Canny edge detector, although starting from a discrete viewpoint and then leading to a set of recursive filters for image smoothing instead of exponential filters or Gaussian filters.

The differential edge detector described below can be seen as a reformulation of Canny's method from the viewpoint of differential invariants computed from a scale-space representation leading to a number of advantages in terms of both theoretical analysis and sub-pixel implementation.

Other first-order methods

For estimating image gradients from the input image or a smoothed version of it, different gradient operators can be applied. The simplest approach is to use central differences:

$$L_x(x, y) = -1/2 \cdot L(x - 1, y) + 0 \cdot L(x, y) + 1/2 \cdot L(x + 1, y)$$
$$L_y(x, y) = -1/2 \cdot L(x, y - 1) + 0 \cdot L(x, y) + 1/2 \cdot L(x, y + 1),$$

corresponding to the application of the following filter masks to the image data:

$$L_x = \begin{bmatrix} -1/2 & 0 & 1/2 \end{bmatrix} * L \quad \text{and} \quad L_y = \begin{bmatrix} +1/2 \\ 0 \\ -1/2 \end{bmatrix} * L.$$

The well-known and earlier Sobel operator is based on the following filters:

$$L_x = \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix} * L \quad \text{and} \quad L_y = \begin{bmatrix} +1 & +2 & +1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} * L.$$

Given such estimates of first- order derivatives, the gradient magnitude is then computed as:

$$|\nabla L| = \sqrt{L_x^2 + L_y^2}$$

while the gradient orientation can be estimated as

$$\theta = \operatorname{atan2}(L_y, L_x).$$

Other first-order difference operators for estimating image gradient have been proposed in the Prewitt operator and Roberts cross.

Thresholding and linking

Once we have computed a measure of edge strength (typically the gradient magnitude), the next stage is to apply a threshold, to decide whether edges are present or not at an image point. The lower the threshold, the more edges will be detected, and the result will be increasingly susceptible to noise and detecting edges of irrelevant features in the image. Conversely a high threshold may miss subtle edges, or result in fragmented edges.

If the edge thresholding is applied to just the gradient magnitude image, the resulting edges will in general be thick and some type of edge thinning post-processing is necessary. For edges detected with non-maximum suppression however, the edge curves are thin by definition and the edge pixels can be linked into edge polygon by an edge linking (edge tracking) procedure. On a discrete grid, the non-maximum suppression stage can be implemented by estimating the gradient direction using first-order derivatives, then rounding off the gradient direction to multiples of 45 degrees, and finally comparing the values of the gradient magnitude in the estimated gradient direction.

A commonly used approach to handle the problem of appropriate thresholds for thresholding is by using thresholding with hysteresis. This method uses multiple thresholds to find edges. We begin by using the upper threshold to find the start of an edge. Once we have a start point, we then trace the path of the edge through the image pixel by pixel, marking an edge whenever we are above the lower threshold. We stop marking our edge only when the value falls below our lower threshold. This approach makes the assumption that edges are likely to be in continuous curves, and allows us to follow a faint section of an edge we have previously seen, without meaning that every noisy pixel in the image is marked down as an edge. Still, however, we have the problem of choosing appropriate thresholding parameters, and suitable thresholding values may vary over the image.

Edge Thinning

Edge thinning is a technique used to remove the unwanted spurious points on the edge of an image. This technique is employed after the image has been filtered for noise (using median, Gaussian filter etc.), the edge operator has been applied (like the ones described above) to detect the edges and after the edges have been smoothed using an appropriate threshold value. This removes all the unwanted points and if applied carefully, results in one pixel thick edge elements.

Advantages: 1) Sharp and thin edges lead to greater efficiency in object recognition. 2) If you are using Hough transforms to detect lines and ellipses then thinning could give much better results. 3) If the edge happens to be boundary of a region then, thinning could easily give the image parameters like perimeter without much algebra.

There are many popular algorithms used to do this, one such is described below:

1) Choose a type of connectivity, like 8, 6 or 4.

2) 8 connectivity is preferred, where all the immediate pixels surrounding a particular pixel are considered.

3) Remove points from North, south, east and west.

4) Do this in multiple passes, i.e. after the north pass, use the same semi processed image in the other passes and so on.

5) Remove a point if:

```
    The point has no neighbors in the North (if you are in the north
pass, and
         respective directions for other passes.)
    The point is not the end of a line.
    The point is isolated.
    Removing the points will not cause to disconnect its neighbors in
any way.
```

6) Else keep the point. The number of passes across direction should be chosen according to the level of accuracy desired.

Second-order approaches to edge detection

Some edge-detection operators are instead based upon second-order derivatives of the intensity. This essentially captures the rate of change in the intensity gradient. Thus, in the ideal continuous case, detection of zero-crossings in the second derivative captures local maxima in the gradient.

The early Marr-Hildreth operator is based on the detection of zero-crossings of the Laplacian operator applied to a Gaussian-smoothed image. It can be shown, however, that this operator will also return false edges corresponding to local minima of the gradient magnitude. Moreover, this operator will give poor localization at curved edges. Hence, this operator is today mainly of historical interest.

Differential edge detection

A more refined second-order edge detection approach which automatically detects edges with sub-pixel accuracy, uses the following *differential approach* of detecting zero-crossings of the second-order directional derivative in the gradient direction:

Following the differential geometric way of expressing the requirement of non-maximum suppression proposed by Lindeberg, let us introduce at every image point a local coordinate system (u,v), with the v-direction parallel to the gradient direction. Assuming that the image has been presmoothed by Gaussian smoothing and a scale-space representation $L(x,y;t)$ at scale t has been computed, we can require that the gradient magnitude of the scale-space representation, which is equal to the first-order directional derivative in the v-direction L_v, should have its first order directional derivative in the v-direction equal to zero

$$\partial_v(L_v) = 0$$

while the second-order directional derivative in the v-direction of L_v should be negative, i.e.,

$$\partial_{vv}(L_v) \leq 0.$$

Written out as an explicit expression in terms of local partial derivatives L_x, L_y ... L_{yyy}, this edge definition can be expressed as the zero-crossing curves of the differential invariant

$$L_v^2 L_{vv} = L_x^2\, L_{xx} + 2\, L_x\, L_y\, L_{xy} + L_y^2\, L_{yy} = 0,$$

that satisfy a sign-condition on the following differential invariant

$$L_v^3 L_{vvv} = L_x^3\, L_{xxx} + 3\, L_x^2\, L_y\, L_{xxy} + 3\, L_x\, L_y^2\, L_{xyy} + L_y^3\, L_{yyy} \leq 0$$

where L_x, L_y ... L_{yyy} denote partial derivatives computed from a scale-space representation L obtained by smoothing the original image with a Gaussian kernel. In this way, the edges will be automatically obtained as continuous curves with subpixel accuracy. Hysteresis thresholding can also be applied to these differential and subpixel edge segments.

In practice, first-order derivative approximations can be computed by central differences as described above, while second-order derivatives can be computed from the scale-space representation L according to:

$$\begin{aligned} L_{xx}(x,y) &= L(x-1,y) - 2L(x,y) + L(x+1,y). \\ L_{xy}(x,y) &= \big(L(x-1,y-1) - L(x-1,y+1) - L(x+1,y-1) + L(x+1,y+1)\big)/4, \\ L_{yy}(x,y) &= L(x,y-1) - 2L(x,y) + L(x,y+1). \end{aligned}$$

corresponding to the following filter masks:

$$L_{xx} = \begin{bmatrix} 1 & -2 & 1 \end{bmatrix} * L \quad \text{and} \quad L_{xy} = \begin{bmatrix} -1/4 & 0 & 1/4 \\ 0 & 0 & 0 \\ 1/4 & 0 & -1/4 \end{bmatrix} * L \quad \text{and} \quad L_{yy} = \begin{bmatrix} 1 \\ -2 \\ 1 \end{bmatrix} * L.$$

Higher-order derivatives for the third-order sign condition can be obtained in an analogous fashion.

Phase congruency based edge detection

A recent development in edge detection techniques takes a frequency domain approach to finding edge locations. Phase congruency (also known as phase coherence) methods attempt to find locations in an image where all sinusoids in the frequency domain are in phase. These locations will generally correspond to the location of a perceived edge, regardless of whether the edge is represented by a large change in intensity in the spatial domain. A key benefit of this technique is that it responds strongly to Mach bands, and avoids false positives typically found around roof edges. A roof edge, is a discontinuity in the first order derivative of a grey-level profile.

Chapter- 6

Segmentation (Image Processing)

In computer vision, **segmentation** refers to the process of partitioning a digital image into multiple segments (sets of pixels, also known as superpixels). The goal of segmentation is to simplify and/or change the representation of an image into something that is more meaningful and easier to analyze. Image segmentation is typically used to locate objects and boundaries (lines, curves, etc.) in images. More precisely, image segmentation is the process of assigning a label to every pixel in an image such that pixels with the same label share certain visual characteristics.

The result of image segmentation is a set of segments that collectively cover the entire image, or a set of contours extracted from the image. Each of the pixels in a region are similar with respect to some characteristic or computed property, such as color, intensity, or texture. Adjacent regions are significantly different with respect to the same characteristic(s).

Applications

Some of the practical applications of image segmentation are:

- Medical imaging
  - Locate tumors and other pathologies
  - Measure tissue volumes
  - Computer-guided surgery
  - Diagnosis
  - Treatment planning
  - Study of anatomical structure
- Locate objects in satellite images (roads, forests, etc.)
- Face recognition
- Fingerprint recognition
- Traffic control systems
- Brake light detection
- Machine vision
- Agricultural imaging – crop disease detection

Several general-purpose algorithms and techniques have been developed for image segmentation. Since there is no general solution to the image segmentation problem, these techniques often have to be combined with domain knowledge in order to effectively solve an image segmentation problem for a problem domain.

Clustering methods

The K-means algorithm is an iterative technique that is used to partition an image into *K* clusters. The basic algorithm is:

1. Pick *K* cluster centers, either randomly or based on some heuristic
2. Assign each pixel in the image to the cluster that minimizes the distance between the pixel and the cluster center
3. Re-compute the cluster centers by averaging all of the pixels in the cluster
4. Repeat steps 2 and 3 until convergence is attained (e.g. no pixels change clusters)

In this case, distance is the squared or absolute difference between a pixel and a cluster center. The difference is typically based on pixel color, intensity, texture, and location, or a weighted combination of these factors. *K* can be selected manually, randomly, or by a heuristic.

This algorithm is guaranteed to converge, but it may not return the optimal solution. The quality of the solution depends on the initial set of clusters and the value of *K*.

In statistics and machine learning, the k-means algorithm is a clustering algorithm to partition n objects into k clusters, where k < n. It is similar to the expectation-maximization algorithm for mixtures of Gaussians in that they both attempt to find the centers of natural clusters in the data. The model requires that the object attributes correspond to elements of a vector space. The objective it tries to achieve is to minimize total intra-cluster variance, or, the squared error function. The k-means clustering was invented in 1956. The most common form of the algorithm uses an iterative refinement heuristic known as Lloyd's algorithm. Lloyd's algorithm starts by partitioning the input points into k initial sets, either at random or using some heuristic data. It then calculates the mean point, or centroid, of each set. It constructs a new partition by associating each point with the closest centroid. Then the centroids are recalculated for the new clusters, and algorithm repeated by alternate application of these two steps until convergence, which is obtained when the points no longer switch clusters (or alternatively centroids are no longer changed). Lloyd's algorithm and k-means are often used synonymously, but in reality Lloyd's algorithm is a heuristic for solving the k-means problem, as with certain combinations of starting points and centroids, Lloyd's algorithm can in fact converge to the wrong answer. Other variations exist, but Lloyd's algorithm has remained popular, because it converges extremely quickly in practice. In terms of performance the algorithm is not guaranteed to return a global optimum. The quality of the final solution depends largely on the initial set of clusters, and may, in practice, be much poorer than the global optimum. Since the algorithm is extremely fast, a common method is to run the algorithm several times and return the best clustering found. A drawback of the k-means

algorithm is that the number of clusters k is an input parameter. An inappropriate choice of k may yield poor results. The algorithm also assumes that the variance is an appropriate measure of cluster scatter.

Compression-based methods

Compression based methods postulate that the optimal segmentation is the one that minimizes, over all possible segmentations, the coding length of the data. The connection between these two concepts is that segmentation tries to find patterns in an image and any regularity in the image can be used to compress it. The method describes each segment by its texture and boundary shape. Each of these components is modeled by a probability distribution function and its coding length is computed as follows:

1. The boundary encoding leverages the fact that regions in natural images tend to have a smooth contour. This prior is used by huffman coding to encode the difference chain code of the contours in an image. Thus, the smoother a boundary is, the shorter coding length it attains.
2. Texture is encoded by lossy compression in a way similar to minimum description length (MDL) principle, but here the length of the data given the model is approximated by the number of samples times the entropy of the model. The texture in each region is modeled by a multivariate normal distribution whose entropy has closed form expression. An interesting property of this model is that the estimated entropy bounds the true entropy of the data from above. This is because among all distributions with a given mean and covariance, normal distribution has the largest entropy. Thus, the true coding length cannot be more than what the algorithm tries to minimize.

For any given segmentation of an image, this scheme yields the number of bits required to encode that image based on the given segmentation. Thus, among all possible segmentations of an image, the goal is to find the segmentation which produces the shortest coding length. This can be achieved by a simple agglomerative clustering method. The distortion in the lossy compression determines the coarseness of the segmentation and its optimal value may differ for each image. This parameter can be estimated heuristically from the contrast of textures in an image. For example, when the textures in an image are similar, such as in camouflage images, stronger sensitivity and thus lower quantization is required.

Histogram-based methods

Histogram-based methods are very efficient when compared to other image segmentation methods because they typically require only one pass through the pixels. In this technique, a histogram is computed from all of the pixels in the image, and the peaks and valleys in the histogram are used to locate the clusters in the image. Color or intensity can be used as the measure.

A refinement of this technique is to recursively apply the histogram-seeking method to clusters in the image in order to divide them into smaller clusters. This is repeated with smaller and smaller clusters until no more clusters are formed.

One disadvantage of the histogram-seeking method is that it may be difficult to identify significant peaks and valleys in the image. In this technique of image classification distance metric and integrated region matching are familiar.

Histogram-based approaches can also be quickly adapted to occur over multiple frames, while maintaining their single pass efficiency. The histogram can be done in multiple fashions when multiple frames are considered. The same approach that is taken with one frame can be applied to multiple, and after the results are merged, peaks and valleys that were previously difficult to identify are more likely to be distinguishable. The histogram can also be applied on a per pixel basis where the information result are used to determine the most frequent color for the pixel location. This approach segments based on active objects and a static environment, resulting in a different type of segmentation useful in Video tracking.

Edge detection

Edge detection is a well-developed field on its own within image processing. Region boundaries and edges are closely related, since there is often a sharp adjustment in intensity at the region boundaries. Edge detection techniques have therefore been used as the base of another segmentation technique.

The edges identified by edge detection are often disconnected. To segment an object from an image however, one needs closed region boundaries.

Region growing methods

The first region growing method was the seeded region growing method. This method takes a set of seeds as input along with the image. The seeds mark each of the objects to be segmented. The regions are iteratively grown by comparing all unallocated neighbouring pixels to the regions. The difference between a pixel's intensity value and the region's mean, δ, is used as a measure of similarity. The pixel with the smallest difference measured this way is allocated to the respective region. This process continues until all pixels are allocated to a region.

Seeded region growing requires seeds as additional input. The segmentation results are dependent on the choice of seeds. Noise in the image can cause the seeds to be poorly placed. Unseeded region growing is a modified algorithm that doesn't require explicit seeds. It starts off with a single region A_1 – the pixel chosen here does not significantly influence final segmentation. At each iteration it considers the neighbouring pixels in the same way as seeded region growing. It differs from seeded region growing in that if the minimum δ is less than a predefined threshold T then it is added to the respective region

A_j. If not, then the pixel is considered significantly different from all current regions A_i and a new region A_{n+1} is created with this pixel.

One variant of this technique, proposed by Haralick and Shapiro (1985), is based on pixel intensities. The mean and scatter of the region and the intensity of the candidate pixel is used to compute a test statistic. If the test statistic is sufficiently small, the pixel is added to the region, and the region's mean and scatter are recomputed. Otherwise, the pixel is rejected, and is used to form a new region.

A special region growing method is called λ-connected segmentation. It is based on pixel intensities and neighborhood linking paths. A degree of connectivity (connectedness) will be calculated based on a path that is formed by pixels. For a certain value of λ, two pixels are called λ-connected if there is a path linking those two pixels and the connectedness of this path is at least λ. λ-connectedness is an equivalence relation.

Partial differential equation-based methods

Using a partial differential equation (PDE)-based method and solving the PDE equation by a numerical scheme, one can segment the image.

Level set methods

Curve propagation is a popular technique in image analysis for object extraction, object tracking, stereo reconstruction, etc. The central idea behind such an approach is to evolve a curve towards the lowest potential of a cost function, where its definition reflects the task to be addressed and imposes certain smoothness constraints. Lagrangian techniques are based on parameterizing the contour according to some sampling strategy and then evolve each element according to image and internal terms. While such a technique can be very efficient, it suffers from various limitations like deciding on the sampling strategy, estimating the internal geometric properties of the curve, changing its topology, addressing problems in higher dimensions, etc. In each case, a partial differential equation (PDE) called the level set equation is solved by finite differences.

The level set method was initially proposed to track moving interfaces by Osher and Sethian in 1988 and has spread across various imaging domains in the late nineties. It can be used to efficiently address the problem of curve/surface/etc. propagation in an implicit manner. The central idea is to represent the evolving contour using a signed function, where its zero level corresponds to the actual contour. Then, according to the motion equation of the contour, one can easily derive a similar flow for the implicit surface that when applied to the zero-level will reflect the propagation of the contour. The level set method encodes numerous advantages: it is implicit, parameter free, provides a direct way to estimate the geometric properties of the evolving structure, can change the topology and is intrinsic. Furthermore, they can be used to define an optimization framework as proposed by Zhao, Merriman and Osher in 1996. Therefore, one can conclude that it is a very convenient framework to address numerous applications of

computer vision and medical image analysis. Furthermore, research into various level set data structures has led to very efficient implementations of this method.

Graph partitioning methods

Graph partitioning methods can effectively be used for image segmentation. In these methods, the image is modeled as a weighted, undirected graph. Usually a pixel or a group of pixels are associated with nodes and edge weights define the (dis)similarity between the neighborhood pixels. The graph (image) is then partitioned according to a criterion designed to model "good" clusters. Each partition of the nodes (pixels) output from these algorithms are considered an object segment in the image. Some popular algorithms of this category are normalized cuts , random walker , minimum cut , isoperimetric partitioning and minimum spanning tree-based segmentation .

Watershed transformation

The watershed transformation considers the gradient magnitude of an image as a topographic surface. Pixels having the highest gradient magnitude intensities (GMIs) correspond to watershed lines, which represent the region boundaries. Water placed on any pixel enclosed by a common watershed line flows downhill to a common local intensity minimum (LIM). Pixels draining to a common minimum form a catch basin, which represents a segment.

Model based segmentation

The central assumption of such an approach is that structures of interest/organs have a repetitive form of geometry. Therefore, one can seek for a probabilistic model towards explaining the variation of the shape of the organ and then when segmenting an image impose constraints using this model as prior. Such a task involves (i) registration of the training examples to a common pose, (ii) probabilistic representation of the variation of the registered samples, and (iii) statistical inference between the model and the image. State of the art methods in the literature for knowledge-based segmentation involve active shape and appearance models, active contours and deformable templates and level-set based methods.

Multi-scale segmentation

Image segmentations are computed at multiple scales in scale-space and sometimes propagated from coarse to fine scales.

Segmentation criteria can be arbitrarily complex and may take into account global as well as local criteria. A common requirement is that each region must be connected in some sense.

One-dimensional hierarchical signal segmentation

Witkin's seminal work in scale space included the notion that a one-dimensional signal could be unambiguously segmented into regions, with one scale parameter controlling the scale of segmentation.

A key observation is that the zero-crossings of the second derivatives (minima and maxima of the first derivative or slope) of multi-scale-smoothed versions of a signal form a nesting tree, which defines hierarchical relations between segments at different scales. Specifically, slope extrema at coarse scales can be traced back to corresponding features at fine scales. When a slope maximum and slope minimum annihilate each other at a larger scale, the three segments that they separated merge into one segment, thus defining the hierarchy of segments.

Image segmentation and primal sketch

There have been numerous research works in this area, out of which a few have now reached a state where they can be applied either with interactive manual intervention (usually with application to medical imaging) or fully automatically. The following is a brief overview of some of the main research ideas that current approaches are based upon.

The nesting structure that Witkin described is, however, specific for one-dimensional signals and does not trivially transfer to higher-dimensional images. Nevertheless, this general idea has inspired several other authors to investigate coarse-to-fine schemes for image segmentation. Koenderink proposed to study how iso-intensity contours evolve over scales and this approach was investigated in more detail by Lifshitz and Pizer. Unfortunately, however, the intensity of image features changes over scales, which implies that it is hard to trace coarse-scale image features to finer scales using iso-intensity information.

Lindeberg studied the problem of linking local extrema and saddle points over scales, and proposed an image representation called the scale-space primal sketch which makes explicit the relations between structures at different scales, and also makes explicit which image features are stable over large ranges of scale including locally appropriate scales for those. Bergholm proposed to detect edges at coarse scales in scale-space and then trace them back to finer scales with manual choice of both the coarse detection scale and the fine localization scale.

Gauch and Pizer studied the complementary problem of ridges and valleys at multiple scales and developed a tool for interactive image segmentation based on multi-scale watersheds. The use of multi-scale watershed with application to the gradient map has also been investigated by Olsen and Nielsen and been carried over to clinical use by Dam Vincken et al. proposed a hyperstack for defining probabilistic relations between image structures at different scales. The use of stable image structures over scales has been furthered by Ahuja and his co-workers into a fully automated system.

More recently, these ideas for multi-scale image segmentation by linking image structures over scales have been picked up by Florack and Kuijper . Bijaoui and Rué associate structures detected in scale-space above a minimum noise threshold into an object tree which spans multiple scales and corresponds to a kind of feature in the original signal. Extracted features are accurately reconstructed using an iterative conjugate gradient matrix method.

Semi-automatic segmentation

In this kind of segmentation, the user outlines the region of interest with the mouse clicks and algorithms are applied so that the path that best fits the edge of the image is shown.

Techniques like Siox, Livewire, Intelligent Scissors or IT-SNAPS are used in this kind of segmentation.

Neural networks segmentation

Neural Network segmentation relies on processing small areas of an image using an artificial neural network or a set of neural networks. After such processing the decision-making mechanism marks the areas of an image accordingly to the category recognized by the neural network. A type of network designed especially for this is the Kohonen map.

Pulse-Coupled Neural Networks (PCNNs) are neural models proposed by modeling a cat's visual cortex and developed for high-performance biomimetic image processing. In 1989, Eckhorn introduced a neural model to emulate the mechanism of cat's visual cortex. The Eckhorn model provided a simple and effective tool for studying small mammal's visual cortex, and was soon recognized as having significant application potential in image processing. In 1994, the Eckhorn model was adapted to be an image processing algorithm by Johnson, who termed this algorithm Pulse-Coupled Neural Network. Over the past decade, PCNNs have been utilized for a variety of image processing applications, including: image segmentation, feature generation, face extraction, motion detection, region growing, noise reduction, and so on. A PCNN is a two-dimensional neural network. Each neuron in the network corresponds to one pixel in an input image, receiving its corresponding pixel's color information (e.g. intensity) as an external stimulus. Each neuron also connects with its neighboring neurons, receiving local stimuli from them. The external and local stimuli are combined in an internal activation system, which accumulates the stimuli until it exceeds a dynamic threshold, resulting in a pulse output. Through iterative computation, PCNN neurons produce temporal series of pulse outputs. The temporal series of pulse outputs contain information of input images and can be utilized for various image processing applications, such as image segmentation and feature generation. Compared with conventional image processing means, PCNNs have several significant merits, including robustness against noise, independence of geometric variations in input patterns, capability of bridging minor intensity variations in input patterns, etc.

Proprietary software

Several proprietary software packages are available for performing image segmentation

- Pac-n-Zoom Color has a proprietary software that color segments over 16 million colors at photographic quality.

- TurtleSeg is a free interactive 3D image segmentation tool. It supports many common medical image file formats and allows the user to export their segmentation as a binary mask or a 3D surface.

Open source software

Several open source software packages are available for performing image segmentation

- ITK - Insight Segmentation and Registration Toolkit (Open Source)
- ITK-SNAP is a GUI tool that combines manual and semi-automatic segmentation with level sets.
- GIMP which includes among other tools SIOX
- VXL
- ImageMagick
- 3DSlicer
- MITK has a program module for manual segmentation
- OpenCV is a computer vision library originally developed by Intel.
- GRASS GIS has the program module i.smap for image segmentation
- Fiji - Fiji is just ImageJ, an image processing package which includes different segmentation plug-ins.
- AForge.NET - an open source C# framework.

There are also software packages available free of charge for academic purposes:

- GemIdent
- CVIPtools
- MegaWave

Benchmarking

Prague On-line Texture Segmentation Benchmark - enables comparing the performance of segmentation methods with the state-of-the-art segmentation methods on a standardized set of textured images

This server allows:

- to obtain customized experimental texture mosaics and their corresponding ground truth,
- to obtain the benchmark texture mosaic set with their corresponding ground truth,
- to evaluate your working segmentation results and compare them with the state of art algorithms,
- to include your algorithm (reference, abstract, benchmark results) into the benchmark database,
- to check single mosaics evaluation details (criteria values and resulted thematic maps),
- to rank segmentation algorithms according to the most common benchmark criteria,
- to obtain LaTeX coded resulting criteria tables.

Computer-generated texture mosaics and benchmarks are composed from the following test image types:

- monospectral textures
- multispectral textures
- BTF (bidirectional texture function) textures BTF Bonn Database
- ALI hyperspectral satellite images Earth Observing 1
- rotation invariant texture set
- scale invariant texture set
- illumination invariant texture set

All generated texture mosaics can be corrupted with additive Gaussian noise, Poisson or salt&pepper noise.

Hold-out trainee sets are supplied in the supervised mode.

Submitted results are stored and used to update the ranking of algorithms based on a selected region-based criterion from the following criteria set:

- region-based (including the sensitivity graphs)

· US - under-segmentation · ME - missed error · NE - noise error
· CS - correct segmentation · OS - over-segmentation

- pixel-wise

· O - omission error · C - commission error,
· CA - class accuracy · CO - recall = correct assignment,
· CC - precision = object accuracy
· I. - type I error · II. - type II error

· RM - root mean square proportion estimation error,
· EA - mean class accuracy estimate
· MS - mapping score · CI - comparison index
· F-measure (weighted harmonic mean of precision and recall) graph,

- consistency measures

· GCE - global consistency error,
· LCE - local consistency error,

- clustering

· dM - Mirkin metric,
· dD - Van Dongen metric,
· dVI - variation of information.

Chapter- 7

Speech Recognition

The display of the Speech Recognition screensaver on a PC, in which the character responds to questions, e.g. "Where are you?" or statements, e.g. "Hello."

Speech recognition (also known as **automatic speech recognition** or **computer speech recognition**) converts spoken words to text. The term "voice recognition" is sometimes used to refer to recognition systems that must be trained to a particular speaker—as is the case for most desktop recognition software. Recognizing the speaker can simplify the task of translating speech.

Speech recognition is a broader solution which refers to technology that can recognize speech without being targeted at single speaker—such as a call system that can recognize arbitrary voices.

Speech recognition applications include voice user interfaces such as voice dialing (e.g., "Call home"), call routing (e.g., "I would like to make a collect call"), domotic appliance control, search (e.g., find a podcast where particular words were spoken), simple data entry (e.g., entering a credit card number), preparation of structured documents (e.g., a radiology report), speech-to-text processing (e.g., word processors or emails), and aircraft (usually termed Direct Voice Input).

History

The first speech recognizer appeared in 1952 and consisted of a device for the recognition of single spoken digits Another early device was the IBM Shoebox, exhibited at the 1964 New York World's Fair.

One of the most notable domains for the commercial application of speech recognition in the United States has been health care and in particular the work of the medical transcriptionist (MT). According to industry experts, at its inception, speech recognition (SR) was sold as a way to completely eliminate transcription rather than make the transcription process more efficient, hence it was not accepted. It was also the case that SR at that time was often technically deficient. Additionally, to be used effectively, it required changes to the ways physicians worked and documented clinical encounters, which many if not all were reluctant to do. The biggest limitation to speech recognition automating transcription, however, is seen as the software. The nature of narrative dictation is highly interpretive and often requires judgment that may be provided by a real human but not yet by an automated system. Another limitation has been the extensive amount of time required by the user and/or system provider to train the software.

A distinction in ASR is often made between "artificial syntax systems" which are usually domain-specific and "natural language processing" which is usually language-specific. Each of these types of application presents its own particular goals and challenges.

Applications

Health care

In the health care domain, even in the wake of improving speech recognition technologies, medical transcriptionists (MTs) have not yet become obsolete. The services provided may be redistributed rather than replaced.

Speech recognition can be implemented in front-end or back-end of the medical documentation process.

Front-End SR is where the provider dictates into a speech-recognition engine, the recognized words are displayed right after they are spoken, and the dictator is responsible for editing and signing off on the document. It never goes through an MT/editor.

Back-End SR or Deferred SR is where the provider dictates into a digital dictation system, and the voice is routed through a speech-recognition machine and the recognized draft document is routed along with the original voice file to the MT/editor, who edits the draft and finalizes the report. Deferred SR is being widely used in the industry currently.

Many Electronic Medical Records (EMR) applications can be more effective and may be performed more easily when deployed in conjunction with a speech-recognition engine. Searches, queries, and form filling may all be faster to perform by voice than by using a keyboard.

Military

High-performance fighter aircraft

Substantial efforts have been devoted in the last decade to the test and evaluation of speech recognition in fighter aircraft. Of particular note is the U.S. program in speech recognition for the Advanced Fighter Technology Integration (AFTI)/F-16 aircraft (F-16 VISTA), and a program in France installing speech recognition systems on Mirage aircraft, and also programs in the UK dealing with a variety of aircraft platforms. In these programs, speech recognizers have been operated successfully in fighter aircraft, with applications including: setting radio frequencies, commanding an autopilot system, setting steer-point coordinates and weapons release parameters, and controlling flight displays.

Working with Swedish pilots flying in the JAS-39 Gripen cockpit, Englund (2004) found recognition deteriorated with increasing G-loads. It was also concluded that adaptation greatly improved the results in all cases and introducing models for breathing was shown to improve recognition scores significantly. Contrary to what might be expected, no effects of the broken English of the speakers were found. It was evident that spontaneous speech caused problems for the recognizer, as could be expected. A restricted vocabulary, and above all, a proper syntax, could thus be expected to improve recognition accuracy substantially.

The Eurofighter Typhoon currently in service with the UK RAF employs a speaker-dependent system, i.e. it requires each pilot to create a template. The system is not used for any safety critical or weapon critical tasks, such as weapon release or lowering of the undercarriage, but is used for a wide range of other cockpit functions. Voice commands are confirmed by visual and/or aural feedback. The system is seen as a major design feature in the reduction of pilot workload, and even allows the pilot to assign targets to himself with two simple voice commands or to any of his wingmen with only five commands.

Speaker independent systems are also being developed and are in testing for The F35 Lightning II (JSF) and the Aermacchi M346 lead in fighter trainer. These systems have produced word accuracies in excess of 98%.

Helicopters

The problems of achieving high recognition accuracy under stress and noise pertain strongly to the helicopter environment as well as to the fighter environment. The acoustic noise problem is actually more severe in the helicopter environment, not only because of the high noise levels but also because the helicopter pilot generally does not wear a facemask, which would reduce acoustic noise in the microphone. Substantial test and evaluation programs have been carried out in the past decade in speech recognition systems applications in helicopters, notably by the U.S. Army Avionics Research and Development Activity (AVRADA) and by the Royal Aerospace Establishment (RAE) in

the UK. Work in France has included speech recognition in the Puma helicopter. There has also been much useful work in Canada. Results have been encouraging, and voice applications have included: control of communication radios; setting of navigation systems; and control of an automated target handover system.

As in fighter applications, the overriding issue for voice in helicopters is the impact on pilot effectiveness. Encouraging results are reported for the AVRADA tests, although these represent only a feasibility demonstration in a test environment. Much remains to be done both in speech recognition and in overall speech recognition technology, in order to consistently achieve performance improvements in operational settings.

Battle management

Battle Management command centres generally require rapid access to and control of large, rapidly changing information databases. Commanders and system operators need to query these databases as conveniently as possible, in an eyes-busy environment where much of the information is presented in a display format. Human-machine interaction by voice has the potential to be very useful in these environments. A number of efforts have been undertaken to interface commercially available isolated-word recognizers into battle management environments. In one feasibility study speech recognition equipment was tested in conjunction with an integrated information display for naval battle management applications. Users were very optimistic about the potential of the system, although capabilities were limited.

Speech understanding programs sponsored by the Defense Advanced Research Projects Agency (DARPA) in the U.S. has focused on this problem of natural speech interface. Speech recognition efforts have focused on a database of continuous speech recognition (CSR), large-vocabulary speech which is designed to be representative of the naval resource management task. Significant advances in the state-of-the-art in CSR have been achieved, and current efforts are focused on integrating speech recognition and natural language processing to allow spoken language interaction with a naval resource management system.

Training air traffic controllers

Training for air traffic controllers (ATC) represents an excellent application for speech recognition systems. Many ATC training systems currently require a person to act as a "pseudo-pilot", engaging in a voice dialog with the trainee controller, which simulates the dialog which the controller would have to conduct with pilots in a real ATC situation. Speech recognition and synthesis techniques offer the potential to eliminate the need for a person to act as pseudo-pilot, thus reducing training and support personnel. In theory, Air controller tasks are also characterized by highly structured speech as the primary output of the controller, hence reducing the difficulty of the speech recognition task should be possible. In practice this is rarely the case. The FAA document 7110.65 details the phrases that should be used by air traffic controllers. While this document gives less than

150 examples of such phrases, the number of phrases supported by one of the simulation vendors speech recognition systems is in excess of 500,000.

The USAF, USMC, US Army,US Navy and FAA as well as a number of international ATC training organizations such as the Royal Australian Air Force and Civil Aviation Authorities in Italy, Brazil, Canada are currently using ATC simulators with speech recognition from a number of different vendors.

Telephony and other domains

ASR in the field of telephony is now commonplace and in the field of computer gaming and simulation is becoming more widespread. Despite the high level of integration with word processing in general personal computing, however, ASR in the field of document production has not seen the expected increases in use.

The improvement of mobile processor speeds made feasible the speech-enabled Symbian and Windows Mobile Smartphones. Speech is used mostly as a part of User Interface, for creating pre-defined or custom speech commands. Leading software vendors in this field are: Microsoft Corporation (Microsoft Voice Command), Digital Syphon (Sonic Extractor), Nuance Communications (Nuance Voice Control), Vito Technology (VITO Voice2Go), Speereo Software (Speereo Voice Translator and SVOX.

Further applications

- Automatic translation;
- Automotive speech recognition (e.g., Ford Sync);
- Telematics (e.g. vehicle Navigation Systems);
- Court reporting (Realtime Voice Writing);
- Hands-free computing: voice command recognition computer user interface;
- Home automation;
- Interactive voice response;
- Mobile telephony, including mobile email;
- Multimodal interaction;
- Pronunciation evaluation in computer-aided language learning applications;
- Robotics;
- Video games, with Tom Clancy's EndWar and Lifeline as working examples;
- Transcription (digital speech-to-text);
- Speech-to-text (transcription of speech into mobile text messages);

Performance

The performance of speech recognition systems is usually specified in terms of accuracy and speed. Accuracy is usually rated with word error rate (WER), whereas speed is measured with the real time factor. Other measures of accuracy include Single Word Error Rate (SWER) and Command Success Rate (CSR).

In 1982, Kurzweil Applied Intelligence and Dragon Systems released speech recognition products. By 1985, Kurzweil's software had a vocabulary of 1,000 words—if uttered one word at a time. Two years later, in 1987, its lexicon reached 20,000 words, entering the realm of human vocabularies, which range from 10,000 to 150,000 words. But recognition accuracy was only 10% in 1993. Two years later, the error rate crossed below 50%. Dragon Systems released "Naturally Speaking" in 1997 which recognized normal human speech. Progress mainly came from improved computer performance and larger source text databases. The Brown Corpus was the first major database available, containing several million words. In 2006, Google published a trillion-word corpus, while Carnegie Mellon University researchers found no significant increase in recognition accuracy.

Algorithms

Both acoustic modeling and language modeling are important parts of modern statistically-based speech recognition algorithms. Hidden Markov models (HMMs) are widely used in many systems. Language modeling has many other applications such as smart keyboard and document classification.

Hidden Markov models

Modern general-purpose speech recognition systems are based on Hidden Markov Models. These are statistical models which output a sequence of symbols or quantities. HMMs are used in speech recognition because a speech signal can be viewed as a piecewise stationary signal or a short-time stationary signal. In a short-time (e.g., 10 milliseconds)), speech can be approximated as a stationary process. Speech can be thought of as a Markov model for many stochastic purposes.

Another reason why HMMs are popular is because they can be trained automatically and are simple and computationally feasible to use. In speech recognition, the hidden Markov model would output a sequence of *n*-dimensional real-valued vectors (with *n* being a small integer, such as 10), outputting one of these every 10 milliseconds. The vectors would consist of cepstral coefficients, which are obtained by taking a Fourier transform of a short time window of speech and decorrelating the spectrum using a cosine transform, then taking the first (most significant) coefficients. The hidden Markov model will tend to have in each state a statistical distribution that is a mixture of diagonal covariance Gaussians which will give a likelihood for each observed vector. Each word, or (for more general speech recognition systems), each phoneme, will have a different output distribution; a hidden Markov model for a sequence of words or phonemes is made by concatenating the individual trained hidden Markov models for the separate words and phonemes.

Described above are the core elements of the most common, HMM-based approach to speech recognition. Modern speech recognition systems use various combinations of a number of standard techniques in order to improve results over the basic approach described above. A typical large-vocabulary system would need context dependency for

the phonemes (so phonemes with different left and right context have different realizations as HMM states); it would use cepstral normalization to normalize for different speaker and recording conditions; for further speaker normalization it might use vocal tract length normalization (VTLN) for male-female normalization and maximum likelihood linear regression (MLLR) for more general speaker adaptation. The features would have so-called delta and delta-delta coefficients to capture speech dynamics and in addition might use heteroscedastic linear discriminant analysis (HLDA); or might skip the delta and delta-delta coefficients and use splicing and an LDA-based projection followed perhaps by heteroscedastic linear discriminant analysis or a global semitied covariance transform (also known as maximum likelihood linear transform, or MLLT). Many systems use so-called discriminative training techniques which dispense with a purely statistical approach to HMM parameter estimation and instead optimize some classification-related measure of the training data. Examples are maximum mutual information (MMI), minimum classification error (MCE) and minimum phone error (MPE).

Decoding of the speech (the term for what happens when the system is presented with a new utterance and must compute the most likely source sentence) would probably use the Viterbi algorithm to find the best path, and here there is a choice between dynamically creating a combination hidden Markov model which includes both the acoustic and language model information, or combining it statically beforehand (the finite state transducer, or FST, approach).

Dynamic time warping (DTW)-based speech recognition

Dynamic time warping is an approach that was historically used for speech recognition but has now largely been displaced by the more successful HMM-based approach. Dynamic time warping is an algorithm for measuring similarity between two sequences which may vary in time or speed. For instance, similarities in walking patterns would be detected, even if in one video the person was walking slowly and if in another they were walking more quickly, or even if there were accelerations and decelerations during the course of one observation. DTW has been applied to video, audio, and graphics – indeed, any data which can be turned into a linear representation can be analyzed with DTW.

A well known application has been automatic speech recognition, to cope with different speaking speeds. In general, it is a method that allows a computer to find an optimal match between two given sequences (e.g. time series) with certain restrictions, i.e. the sequences are "warped" non-linearly to match each other. This sequence alignment method is often used in the context of hidden Markov models.

Further information

Popular speech recognition conferences held each year or two include SpeechTEK and SpeechTEK Europe, ICASSP, Eurospeech/ICSLP (now named Interspeech) and the IEEE ASRU. Conferences in the field of Natural language processing, such as ACL, NAACL, EMNLP, and HLT, are beginning to include papers on speech processing. Important journals include the IEEE Transactions on Speech and Audio Processing (now named IEEE Transactions on Audio, Speech and Language Processing), Computer Speech and Language, and Speech Communication. Books like "Fundamentals of Speech Recognition" by Lawrence Rabiner can be useful to acquire basic knowledge but may not be fully up to date (1993). Another good source can be "Statistical Methods for Speech Recognition" by Frederick Jelinek and "Spoken Language Processing (2001)" by Xuedong Huang etc. More up to date is "Computer Speech", by Manfred R. Schroeder, second edition published in 2004. The recently updated textbook of "Speech and Language Processing (2008)" by Jurafsky and Martin presents the basics and the state of the art for ASR. A good insight into the techniques used in the best modern systems can be gained by paying attention to government sponsored evaluations such as those organised by DARPA (the largest speech recognition-related project ongoing as of 2007 is the GALE project, which involves both speech recognition and translation components).

In terms of freely available resources, Carnegie Mellon University's SPHINX toolkit is one place to start to both learn about speech recognition and to start experimenting. Another resource (free as in free beer, not free software) is the HTK book (and the accompanying HTK toolkit). The AT&T libraries GRM library, and DCD library are also general software libraries for large-vocabulary speech recognition.

A useful review of the area of robustness in ASR is provided by Junqua and Haton (1995).

People with disabilities

People with disabilities can benefit from speech recognition programs. Speech recognition is especially useful for people who have difficulty using their hands, ranging from mild repetitive stress injuries to involved disabilities that preclude using conventional computer input devices. In fact, people who used the keyboard a lot and developed RSI became an urgent early market for speech recognition. Speech recognition is used in deaf telephony, such as voicemail to text, relay services, and captioned telephone. Individuals with learning disabilities who have problems with thought-to-paper communication (essentially they think of an idea but it is processed incorrectly causing it to end up differently on paper) can benefit from the software.

Current research and funding

Measuring progress in speech recognition performance is difficult and controversial. Some speech recognition tasks are much more difficult than others. Word error rates on some tasks are less than 1%. On others they can be as high as 50%. Sometimes it even appears that performance is going backwards as researchers undertake harder tasks that have higher error rates.

Because progress is slow and is difficult to measure, there is some perception that performance has plateaued and that funding has dried up or shifted priorities. Such perceptions are not new. In 1969, John Pierce wrote an open letter that did cause much funding to dry up for several years. In 1993 there was a strong feeling that performance had plateaued and there were workshops dedicated to the issue. However, in the 1990s funding continued more or less uninterrupted and performance continued slowly but steadily to improve.

For the past thirty years, speech recognition research has been characterized by the steady accumulation of small incremental improvements. There has also been a trend continually to change focus to more difficult tasks due both to progress in speech recognition performance and to the availability of faster computers. In particular, this shifting to more difficult tasks has characterized DARPA funding of speech recognition since the 1980s. In the last decade it has continued with the EARS project, which undertook recognition of Mandarin and Arabic in addition to English, and the GALE project, which focused solely on Mandarin and Arabic and required translation simultaneously with speech recognition.

Commercial research and other academic research also continue to focus on increasingly difficult problems. One key area is to improve robustness of speech recognition performance, not just robustness against noise but robustness against any condition that causes a major degradation in performance. Another key area of research is focused on an opportunity rather than a problem. This research attempts to take advantage of the fact that in many applications there is a large quantity of speech data available, up to millions of hours. It is too expensive to have humans transcribe such large quantities of speech, so the research focus is on developing new methods of machine learning that can effectively utilize large quantities of unlabeled data. Another area of research is better understanding of human capabilities and to use this understanding to improve machine recognition performance.

Chapter- 8

Data Compression

In computer science and information theory, **data compression**, **source coding** or **bit-rate reduction** is the process of encoding information using fewer bits than the original representation would use.

Compression is useful because it helps reduce the consumption of expensive resources, such as hard disk space or transmission bandwidth. On the downside, compressed data must be decompressed to be used, and this extra processing may be detrimental to some applications. For instance, a compression scheme for video may require expensive hardware for the video to be decompressed fast enough to be viewed as it is being decompressed (the option of decompressing the video in full before watching it may be inconvenient, and requires storage space for the decompressed video). The design of data compression schemes therefore involves trade-offs among various factors, including the degree of compression, the amount of distortion introduced (if using a lossy compression scheme), and the computational resources required to compress and uncompress the data.

Lossless versus lossy compression

Lossless compression algorithms usually exploit statistical redundancy in such a way as to represent the sender's data more concisely without error. Lossless compression is possible because most real-world data has statistical redundancy. For example, in English text, the letter 'e' is much more common than the letter 'z', and the probability that the letter 'q' will be followed by the letter 'z' is very small. Another kind of compression, called *lossy data compression* or perceptual coding, is possible if some loss of fidelity is acceptable. Generally, a lossy data compression will be guided by research on how people perceive the data in question. For example, the human eye is more sensitive to subtle variations in luminance than it is to variations in color. JPEG image compression works in part by "rounding off" some of this less-important information. Lossy data compression provides a way to obtain the best fidelity for a given amount of compression. In some cases, *transparent* (unnoticeable) compression is desired; in other cases, fidelity is sacrificed to reduce the amount of data as much as possible.

Lossless compression schemes are reversible so that the original data can be reconstructed, while lossy schemes accept some loss of data in order to achieve higher compression.

However, lossless data compression algorithms will always fail to compress some files; indeed, any compression algorithm will necessarily fail to compress any data containing no discernible patterns. Attempts to compress data that has been compressed already will therefore usually result in an expansion, as will attempts to compress all but the most trivially encrypted data.

In practice, lossy data compression will also come to a point where compressing again does not work, although an extremely lossy algorithm, like for example always removing the last byte of a file, will always compress a file up to the point where it is empty.

An example of lossless vs. lossy compression is the following string:

    25.888888888

This string can be compressed as:

    25.8

Interpreted as, "twenty five point 9 eights", the original string is perfectly recreated, just written in a smaller form. In a lossy system, using

    26

instead, the exact original data is lost, at the benefit of a shorter representation.

Example algorithms and applications

The above is a very simple example of run-length encoding, wherein large runs of consecutive identical data values are replaced by a simple code with the data value and length of the run. This is an example of lossless data compression. It is often used to optimize disk space on office computers, or better use the connection bandwidth in a computer network. For symbolic data such as spreadsheets, text, executable programs, etc., losslessness is essential because changing even a single bit cannot be tolerated (except in some limited cases).

For visual and audio data, some loss of quality can be tolerated without losing the essential nature of the data. By taking advantage of the limitations of the human sensory system, a great deal of space can be saved while producing an output which is nearly indistinguishable from the original. These lossy data compression methods typically offer a three-way tradeoff between compression speed, compressed data size and quality loss.

Lossy

Lossy image compression is used in digital cameras, to increase storage capacities with minimal degradation of picture quality. Similarly, DVDs use the lossy MPEG-2 Video codec for video compression.

In lossy audio compression, methods of psychoacoustics are used to remove non-audible (or less audible) components of the signal. Compression of human speech is often performed with even more specialized techniques, so that "speech compression" or "voice coding" is sometimes distinguished as a separate discipline from "audio compression". Different audio and speech compression standards are listed under audio codecs. Voice compression is used in Internet telephony for example, while audio compression is used for CD ripping and is decoded by audio players.

Lossless

The Lempel–Ziv (LZ) compression methods are among the most popular algorithms for lossless storage. DEFLATE is a variation on LZ which is optimized for decompression speed and compression ratio, therefore compression can be slow. DEFLATE is used in PKZIP, gzip and PNG. LZW (Lempel–Ziv–Welch) is used in GIF images. Also noteworthy are the LZR (LZ–Renau) methods, which serve as the basis of the Zip method. LZ methods utilize a table-based compression model where table entries are substituted for repeated strings of data. For most LZ methods, this table is generated dynamically from earlier data in the input. The table itself is often Huffman encoded (e.g. SHRI, LZX). A current LZ-based coding scheme that performs well is LZX, used in Microsoft's CAB format.

The very best modern losless compressors use probabilistic models, such as prediction by partial matching. The Burrows–Wheeler transform can also be viewed as an indirect form of statistical modelling.

In a further refinement of these techniques, statistical predictions can be coupled to an algorithm called arithmetic coding. Arithmetic coding, invented by Jorma Rissanen, and turned into a practical method by Witten, Neal, and Cleary, achieves superior compression to the better-known Huffman algorithm, and lends itself especially well to adaptive data compression tasks where the predictions are strongly context-dependent. Arithmetic coding is used in the bilevel image-compression standard JBIG, and the document-compression standard DjVu. The text *entry* system, Dasher, is an inverse-arithmetic-coder.

Theory

The theoretical background of compression is provided by information theory (which is closely related to algorithmic information theory) for lossless compression, and by rate–distortion theory for lossy compression. These fields of study were essentially created by Claude Shannon, who published fundamental papers on the topic in the late 1940s and

early 1950s. Cryptography and coding theory are also closely related. The idea of data compression is deeply connected with statistical inference.

Many lossless data compression systems can be viewed in terms of a four-stage model. Lossy data compression systems typically include even more stages, including, for example, prediction, frequency transformation, and quantization.

Machine learning

There is a close connection between machine learning and compression: a system that predicts the posterior probabilities of a sequence given its entire history can be used for optimal data compression (by using arithmetic coding on the output distribution), while an optimal compressor can be used for prediction (by finding the symbol that compresses best, given the previous history). This equivalence has been used as justification for data compression as a benchmark for "general intelligence".

Data differencing

Data compression can be seen as a special case of data differencing – data differencing consists of producing a *difference* given a *source* and a *target*, with patching producing a *target* given a *source* and a *difference,* while data compression consists of producing a compressed file given a target, and decompression consists of producing a target given only a compressed file. Thus, one can consider data compression as data differencing with empty source data, the compressed file corresponding to a "difference from nothing". This is the same as considering absolute entropy (corresponding to data compression) as a special case of relative entropy (corresponding to data differencing) with no initial data.

When one wishes to emphasize the connection, one may use the term *differential compression* to refer to data differencing.

Chapter- 9

Lossless Data Compression

Lossless data compression is a class of data compression algorithms that allows the exact original data to be reconstructed from the compressed data. The term *lossless* is in contrast to lossy data compression, which only allows an approximation of the original data to be reconstructed, in exchange for better compression rates.

Lossless data compression is used in many applications. For example, it is used in the ZIP file format and in the Unix tool gzip. It is also often used as a component within lossy data compression technologies (e.g. lossless mid/side joint stereo preprocessing by the LAME MP3 encoder and other lossy audio encoders).

Lossless compression is used in cases where it is important that the original and the decompressed data be identical, or where deviations from the original data could be deleterious. Typical examples are executable programs, text documents and source code. Some image file formats, like PNG or GIF, use only lossless compression, while others like TIFF and MNG may use either lossless or lossy methods. Lossless audio formats are most often used for archiving or production purposes, with smaller lossy audio files being typically used on portable players and in other cases where storage space is limited and/or exact replication of the audio is unnecessary.

Lossless compression techniques

Most lossless compression programs do two things in sequence: the first step generates a *statistical model* for the input data, and the second step uses this model to map input data to bit sequences in such a way that "probable" (e.g. frequently encountered) data will produce shorter output than "improbable" data.

The primary encoding algorithms used to produce bit sequences are Huffman coding (also used by DEFLATE) and arithmetic coding. Arithmetic coding achieves compression rates close to the best possible for a particular statistical model, which is given by the information entropy, whereas Huffman compression is simpler and faster but produces poor results for models that deal with symbol probabilities close to 1.

There are two primary ways of constructing statistical models: in a *static* model, the data is analyzed and a model is constructed, then this model is stored with the compressed

data. This approach is simple and modular, but has the disadvantage that the model itself can be expensive to store, and also that it forces a single model to be used for all data being compressed, and so performs poorly on files containing heterogeneous data. *Adaptive* models dynamically update the model as the data is compressed. Both the encoder and decoder begin with a trivial model, yielding poor compression of initial data, but as they learn more about the data performance improves. Most popular types of compression used in practice now use adaptive coders.

Lossless compression methods may be categorized according to the type of data they are designed to compress. While, in principle, any general-purpose lossless compression algorithm (*general-purpose* meaning that they can compress any bitstring) can be used on any type of data, many are unable to achieve significant compression on data that are not of the form for which they were designed to compress. Many of the lossless compression techniques used for text also work reasonably well for indexed images.

Text

Statistical modeling algorithms for text (or text-like binary data such as executables) include:

- Context Tree Weighting method (CTW)
- Burrows-Wheeler transform (block sorting preprocessing that makes compression more efficient)
- LZ77 (used by DEFLATE)
- LZW
- PPMd

Multimedia

Techniques that take advantage of the specific characteristics of images such as the common phenomenon of contiguous 2-D areas of similar tones. Every pixel but the first is replaced by the difference to its left neighbor. This leads to small values having a much higher probability than large values. This is often also applied to sound files and can compress files which contain mostly low frequencies and low volumes. For images this step can be repeated by taking the difference to the top pixel, and then in videos the difference to the pixel in the next frame can be taken.

A hierarchical version of this technique takes neighboring pairs of data points, stores their difference and sum, and on a higher level with lower resolution continues with the sums. This is called discrete wavelet transform. JPEG2000 additionally uses data points from other pairs and multiplication factors to mix then into the difference. These factors have to be integers so that the result is an integer under all circumstances. So the values are increased, increasing file size, but hopefully the distribution of values is more peaked.

The adaptive encoding uses the probabilities from the previous sample in sound encoding, from the left and upper pixel in image encoding, and additionally from the

previous frame in video encoding. In the wavelet transformation the probabilities are also passed through the hierarchy.

Historical legal issues

Many of these methods are implemented in open-source and proprietary tools, particularly LZW and its variants. Some algorithms are patented in the USA and other countries and their legal usage requires licensing by the patent holder. Because of patents on certain kinds of LZW compression, and in particular licensing practices by patent holder Unisys that many developers considered abusive, some open source proponents encouraged people to avoid using the Graphics Interchange Format (GIF) for compressing image files in favor of Portable Network Graphics PNG, which combines the LZ77-based deflate algorithm with a selection of domain-specific prediction filters. However, the patents on LZW have expired on June 20 2003.

Many of the lossless compression techniques used for text also work reasonably well for indexed images, but there are other techniques that do not work for typical text that are useful for some images (particularly simple bitmaps), and other techniques that take advantage of the specific characteristics of images (such as the common phenomenon of contiguous 2-D areas of similar tones, and the fact that color images usually have a preponderance of a limited range of colors out of those representable in the color space).

As mentioned previously, lossless sound compression is a somewhat specialised area. Lossless sound compression algorithms can take advantage of the repeating patterns shown by the wave-like nature of the data – essentially using models to predict the "next" value and encoding the (hopefully small) difference between the expected value and the actual data. If the difference between the predicted and the actual data (called the "error") tends to be small, then certain difference values (like 0, +1, -1 etc. on sample values) become very frequent, which can be exploited by encoding them in few output bits.

It is sometimes beneficial to compress only the differences between two versions of a file (or, in video compression, of an image). This is called delta compression (from the Greek letter Δ which is commonly used in mathematics to denote a difference), but the term is typically only used if both versions are meaningful outside compression and decompression. For example, while the process of compressing the error in the above-mentioned lossless audio compression scheme could be described as delta compression from the approximated sound wave to the original sound wave, the approximated version of the sound wave is not meaningful in any other context.

Lossless compression methods

By operation of the pigeonhole principle, no lossless compression algorithm can efficiently compress all possible data, and completely random data streams cannot be compressed. For this reason, many different algorithms exist that are designed either with a specific type of input data in mind or with specific assumptions about what kinds of redundancy the uncompressed data are likely to contain.

Some of the most common lossless compression algorithms are listed below.

General purpose

- Run-length encoding – a simple scheme that provides good compression of data containing lots of runs of the same value.
- LZW – used by GIF images and compress among applications
- DEFLATE – used by gzip, modern versions of zip and as part of the compression process of PNG, PPP, HTTP, SSH
- bzip2 - a successor to deflate, slower but with higher compression
- Lempel–Ziv–Markov chain algorithm (LZMA) - used by 7zip, xz, and other programs; higher compression than bzip2
- Lempel–Ziv–Oberhumer (LZO) - designed for compression/decompression speed at the expense of compression ratios
- Statistical Lempel Ziv - a combination of statistical method and dictionary-based method; better compression ratio than using single method.

Audio

- Waveform audio format - WAV
- Free Lossless Audio Codec – FLAC
- Apple Lossless – ALAC (Apple Lossless Audio Codec)
- apt-X – Lossless
- Adaptive Transform Acoustic Coding – ATRAC
- Audio Lossless Coding – also known as MPEG-4 ALS
- MPEG-4 SLS – also known as HD-AAC
- Direct Stream Transfer – DST
- Dolby TrueHD
- DTS-HD Master Audio
- Meridian Lossless Packing – MLP
- Monkey's Audio – Monkey's Audio APE
- OptimFROG
- RealPlayer – RealAudio Lossless
- Shorten – SHN
- TTA – True Audio Lossless
- WavPack – WavPack lossless
- WMA Lossless – Windows Media Lossless

Graphics

- ILBM – (lossless RLE compression of Amiga IFF images)
- JBIG2 – (lossless or lossy compression of B&W images)
- JPEG-LS – (lossless/near-lossless compression standard)
- JPEG 2000 – (includes lossless compression method, as proven by Sunil Kumar, Prof San Diego State University)

- JPEG XR – formerly *WMPhoto* and *HD Photo*, includes a lossless compression method
- PGF – Progressive Graphics File (lossless or lossy compression)
- PNG – Portable Network Graphics
- TIFF – Tagged Image File Format
- Gifsicle (GPL) – Optimize gif files
- Jpegoptim (GPL) – Optimize jpeg files

3D Graphics

- OpenCTM – Lossless compression of 3D triangle meshes

Video

- Animation codec
- CorePNG
- Dirac – Has a *lossless* mode.
- FFV1
- JPEG 2000
- Huffyuv
- Lagarith
- MSU Lossless Video Codec
- SheerVideo

Cryptography

Cryptosystems often compress data before encryption for added security; compression prior to encryption helps remove redundancies and patterns that might facilitate cryptanalysis. However, many ordinary lossless compression algorithms introduce predictable patterns (such as headers, wrappers, and tables) into the compressed data that may actually make cryptanalysis easier. Therefore, cryptosystems often incorporate specialized compression algorithms specific to the cryptosystem—or at least demonstrated or widely held to be cryptographically secure—rather than standard compression algorithms that are efficient but provide potential opportunities for cryptanalysis.

Executables

Self-extracting executables contain a compressed application and a decompressor. When executed, the decompressor transparently decompresses and runs the original application. This is especially often used in demo coding, where competitions are held for demos with strict size limits, as small as 1k. This type of compression is not strictly limited to binary executables, but can also be applied to scripts, such as JavaScript.

Lossless compression benchmarks

Lossless compression algorithms and their implementations are routinely tested in head-to-head benchmarks. There are a number of better-known compression benchmarks. Some benchmarks cover only the compression ratio, so winners in these benchmark may be unsuitable for everyday use due to the slow speed of the top performers. Another drawback of some benchmarks is that their data files are known, so some program writers may optimize their programs for best performance on a particular data set. The winners on these benchmarks often come from the class of context-mixing compression software. The benchmarks listed in the 5th edition of the *Handbook of Data Compression* (Springer, 2009) are:

- The Maximum Compression benchmark, started in 2003 and frequently updated, includes over 150 programs. Maintained by Werner Bergmans, it tests on a variety of data sets, including text, images and executable code. Two types of results are reported: single file compression (SFS) and multiple file compression (MFC). Not surprisingly, context mixing programs often wins here; programs from the PAQ series and WinRK often are in the top. The site also has a list of pointers to other benchmarks.
- UCLC (the ultimate command-line compressors) benchmark by Johan de Bock is another actively maintained benchmark including over 100 programs. The winners in most test usually are PAQ programs and WinRK, with the exception of lossless audio encoding and grayscale image compression where some specialized algorithms shine.
- Squeeze Chart by Stephan Busch is another frequently updated site.
- The EmilCont benchmarks by Berto Destasio are somewhat outdated having been most recently updated in 2004. A distinctive feature is that the data set is not made public in order to prevent optimizations targeting it specifically. Nevertheless, the best ratio winner are again the PAQ family, SLIM and WinRK.
- The Archive Comparison Test (ACT) by Jeff Gilchrist included 162 DOS/Windows and 8 Macintosh lossless compression programs, but it was last updated in 2002.
- The Art Of Lossless Data Compression by Alexander Ratushnyak includes provides similar test performed in 2003.

Matt Mahoney, in his February 2010 edition of the free booklet *Data Compression Explained*, additionally lists the following:

- The Calgary Corpus dating back to 1987 is no longer widely used due to its small size, although Leonid A. Broukhis still maintains the The Calgary Corpus Compression Challenge, which started in 1996.
- The Generic Compression Benchmark, maintained by Mahoney himself, test compression on random data.
- Sami Runsas (author of NanoZip) maintains Compression Ratings, a benchmark similar to Maximum Compression multiple file test, but with minimum speed requirements. It also offers a calculator that allows the user to weight the importance of speed and compression ratio. The top programs here are fairly

different due to speed requirement. In January 2010, the top programs were NanoZip followed by FreeArc, CCM, flashzip, and 7-Zip.
- The Monster of Compression benchmark by N. F. Antonio tests compression on 1Gb of public data with a 40 minute time limit. As of Dec. 20, 2009 the top ranked archiver is NanoZip 0.07a and the top ranked single file compressor is ccmx 1.30c, both context mixing.

Limitations

Lossless data compression algorithms cannot guarantee compression for all input data sets. In other words, for any (lossless) data compression algorithm, there will be an input data set that does not get smaller when processed by the algorithm. This is easily proven with elementary mathematics using a counting argument, as follows:

- Assume that each file is represented as a string of bits of some arbitrary length.
- Suppose that there is a compression algorithm that transforms every file into a distinct file which is no longer than the original file, and that at least one file will be compressed into something that is shorter than itself.
- Let M be the least number such that there is a file F with length M bits that compresses to something shorter. Let N be the length (in bits) of the compressed version of F.
- Because $N < M$, **every** file of length N keeps its size during compression. There are 2^N such files. Together with F, this makes $2^N + 1$ files which all compress into one of the 2^N files of length N.
- But 2^N is smaller than $2^N + 1$, so by the pigeonhole principle there must be some file of length N which is simultaneously the output of the compression function on two different inputs. That file cannot be decompressed reliably (which of the two originals should that yield?), which contradicts the assumption that the algorithm was lossless.
- We must therefore conclude that our original hypothesis (that the compression function makes no file longer) is necessarily untrue.

Any lossless compression algorithm that makes some files shorter must necessarily make some files longer, but it is not necessary that those files become *very much* longer. Most practical compression algorithms provide an "escape" facility that can turn off the normal coding for files that would become longer by being encoded. Then the only increase in size is a few bits to tell the decoder that the normal coding has been turned off for the entire input. For example, DEFLATE compressed files never need to grow by more than 5 bytes per 65,535 bytes of input.

In fact, if we consider files of length N, if all files were equally probable, then for any lossless compression that reduces the size of some file, the expected length of a compressed file (averaged over all possible files of length N) must necessarily be *greater* than N. So if we know nothing about the properties of the data we are compressing, we might as well not compress it at all. A lossless compression algorithm is useful only when

we are more likely to compress certain types of files than others; then the algorithm could be designed to compress those types of data better.

Thus, the main lesson from the argument is not that one risks big losses, but merely that one cannot always win. To choose an algorithm always means implicitly to select a *subset* of all files that will become usefully shorter. This is the theoretical reason why we need to have different compression algorithms for different kinds of files: there cannot be any algorithm that is good for all kinds of data.

The "trick" that allows lossless compression algorithms, used on the type of data they were designed for, to consistently compress such files to a shorter form is that the files the algorithms are designed to act on all have some form of easily-modeled redundancy that the algorithm is designed to remove, and thus belong to the subset of files that that algorithm can make shorter, whereas other files would not get compressed or even get bigger. Algorithms are generally quite specifically tuned to a particular type of file: for example, lossless audio compression programs do not work well on text files, and *vice versa*.

In particular, files of random data cannot be consistently compressed by any conceivable lossless data compression algorithm: indeed, this result is used to *define* the concept of randomness in algorithmic complexity theory.

An algorithm that is asserted to be able to losslessly compress any data stream is provably impossible. While there have been many claims through the years of companies achieving "perfect compression" where an arbitrary number N of random bits can always be compressed to N-1 bits, these kinds of claims can be safely discarded without even looking at any further details regarding the purported compression scheme. Such an algorithm contradicts fundamental laws of mathematics because, if it existed, it could be applied repeatedly to losslessly reduce any file to length 0. Allegedly "perfect" compression algorithms are usually called derisively "magic" compression algorithms.

On the other hand, it has also been proven that there is no algorithm to determine whether a file is incompressible in the sense of Kolmogorov complexity; hence, given any particular file, even if it appears random, it's possible that it may be significantly compressed, even including the size of the decompressor. An example is the digits of the mathematical constant *pi*, which appear random but can be generated by a very small program. However, even though it cannot be determined whether a particular file is incompressible, a simple theorem about incompressible strings shows that over 99% of files of any given length cannot be compressed by more than one byte (including the size of the decompressor).

Mathematical background

Any compression algorithm can be viewed as a function that maps sequences of units (normally octets) into other sequences of the same units. Compression is successful if the resulting sequence is shorter than the original sequence plus the map needed to

decompress it. In order for a compression algorithm to be considered lossless, there needs to exist a reverse mapping from compressed bit sequences to original bit sequences; that is to say, the compression method would need to encapsulate a bijection between "plain" and "compressed" bit sequences.

The sequences of length N or less are clearly a strict superset of the sequences of length N-1 or less. It follows that there are more sequences of length N or less than there are sequences of length N-1 or less. It therefore follows from the pigeonhole principle that it is not possible to map every sequence of length N or less to a unique sequence of length N-1 or less. Therefore it is not possible to produce an algorithm that reduces the size of *every possible* input sequence.

Psychological background

Most everyday files are relatively 'sparse' in an information entropy sense, and thus, most lossless algorithms a layperson is likely to apply on regular files compress them relatively well. This may, through misapplication of intuition, lead some individuals to conclude that a well-designed compression algorithm can compress *any* input, thus, constituting a magic compression algorithm.

Points of application in real compression theory

Real compression algorithm designers accept that streams of high information entropy cannot be compressed, and accordingly, include facilities for detecting and handling this condition. An obvious way of detection is applying a raw compression algorithm and testing if its output is smaller than its input. Sometimes, detection is made by heuristics; for example, a compression application may consider files whose names end in ".zip", ".arj" or ".lha" uncompressible without any more sophisticated detection. A common way of handling this situation is quoting input, or uncompressible parts of the input in the output, minimising the compression overhead. For example, the zip data format specifies the 'compression method' of 'Stored' for input files that have been copied into the archive verbatim.

The Million Random Number Challenge

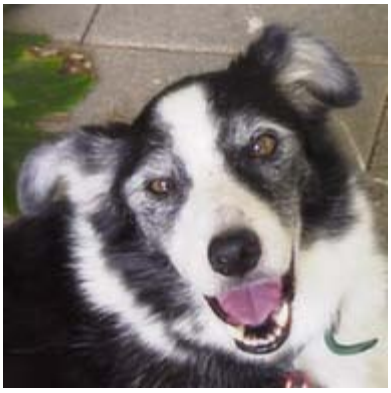
Mark Nelson, frustrated over many cranks trying to claim having invented a magic compression algorithm appearing in comp.compression, has constructed a 415,241 byte binary file () of highly entropic content, and issued a public challenge of $100 to anyone to write a program that, together with its input, would be smaller than his provided binary data yet be able to reconstitute ("decompress") it without error.

The FAQ for the comp.compression newsgroup contains a challenge by Mike Goldman offering $5,000 for a program that can compress random data. Patrick Craig took up the challenge, but rather than compressing the data, he split it up into separate files all of which ended in the number '5' which was not stored as part of the file. Omitting this character allowed the resulting files (plus, in accordance with the rules, the size of the

program that reassembled them) to be smaller than the original file. However, no actual compression took place, and the information stored in the names of the files was necessary in order to reassemble them in the correct order into the original file, and this information was not taken into account in the file size comparison. The files themselves are thus not sufficient to reconstitute the original file; the file names are also necessary. A full history of the event, including discussion on whether or not the challenge was technically met, is on Patrick Craig's web site.

Chapter- 10

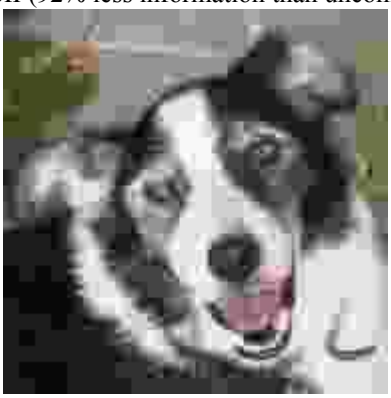
Lossy Compression



Original Image (lossless PNG, 60.1 KB size) — uncompressed is 108.5 KB

Low compression (84% less information than uncompressed PNG, 9.37 KB)

Medium compression (92% less information than uncompressed PNG, 4.82 KB)



High compression (98% less information than uncompressed PNG, 1.14 KB)

In information technology, **"lossy" compression** is a data encoding method which compresses data by discarding (losing) some of it. The procedure aims to minimise the amount of data that needs to be held, handled, and/or transmitted by a computer. The different versions of the photo of the dog at the right demonstrate how much data can be dispensed with, and how the pictures become progressively coarser as the data that made up the original one is discarded (lost). Typically, a substantial amount of data can be discarded before the result is sufficiently degraded to be noticed by the user.

Lossy compression is most commonly used to compress multimedia data (audio, video, still images), especially in applications such as streaming media and internet telephony. By contrast, lossless compression is required for text and data files, such as bank records, text articles, etc. In many cases it is advantageous to make a master lossless file which can then be used to produce compressed files for different purposes; for example a multi-megabyte file can be used at full size to produce a full-page advertisement in a glossy magazine, and a 10 kilobyte lossy copy made for a small image on a web page.

Lossy and lossless compression

It is possible to compress many types of digital data in a way which reduces the size of a computer file needed to store it or the bandwidth needed to stream it, with no loss of the full information contained in the original file. A picture, for example, is converted to a digital file by considering it to be an array of dots, and specifying the color and brightness of each dot. If the picture contains an area of the same color, it can be compressed without loss by saying "200 red dots" instead of "red dot, red dot, ...(197 more times)..., red dot".

The original contains a certain amount of information; there is a lower limit to the size of file that can carry all the information. As an intuitive example, most people know that a compressed ZIP file is smaller than the original file; but repeatedly compressing the file will not reduce the size to nothing, and will in fact usually increase the size.

In many cases files or data streams contain more information than is needed for a particular purpose. For example, a picture may have more detail than the eye can distinguish when reproduced at the largest size intended; an audio file does not need a lot of fine detail during a very loud passage. Developing lossy compression techniques as closely matched to human perception as possible is a complex task. In some cases the ideal is a file which provides exactly the same perception as the original, with as much digital information as possible removed; in other cases perceptible loss of quality is considered a valid trade-off for the reduced data size.

Transform coding

More generally, lossy compression can be thought of as an application of *transform coding* – in the case of multimedia data, *perceptual coding:* it transforms the raw data to a domain that more accurately reflects the information content. For example, rather than expressing a sound file as the amplitude levels over time, one may express it as the frequency spectrum over time, which corresponds more accurately to human audio perception.

While data reduction (compression, be it lossy or lossless) is a main goal of transform coding, it also allows other goals: one may represent data more accurately for the original amount of space – for example, in principle, if one starts with an analog or high-resolution digital master, an MP3 file of a given bitrate (e.g. 320 kbit/s) should provide a better representation than a raw uncompressed audio in WAV or AIFF file of the same bitrate. (Uncompressed audio can get lower bitrate only by lowering sampling frequency and/or sampling resolution.) Further, a transform coding may provide a better domain for manipulating or otherwise editing the data – for example, equalization of audio is most naturally expressed in the frequency domain (boost the bass, for instance) rather than in the raw time domain.

From this point of view, perceptual encoding is not essentially about *discarding* data, but rather about a *better representation* of data.

Another use is for backward compatibility and graceful degradation: in color television, encoding color via a luminance-chrominance transform domain (such as YUV) means that black-and-white sets display the luminance, while ignoring the color information.

Another example is chroma subsampling: the use of color spaces such as YIQ, used in NTSC, allow one to reduce the resolution on the components to accord with human perception – humans have highest resolution for black-and-white (luma), lower resolution for mid-spectrum colors like yellow and green, and lowest for red and blues – thus NTSC displays approximately 350 pixels of luma per scanline, 150 pixels of yellow vs. green, and 50 pixels of blue vs. red, which are proportional to human sensitivity to each component.

Information loss

Lossy compression formats suffer from generation loss: repeatedly compressing and decompressing the file will cause it to progressively lose quality. This is in contrast with lossless data compression, where data will not be lost via the use of such a procedure.

Information-theoretical foundations for lossy data compression are provided by rate-distortion theory. Much like the use of probability in optimal coding theory, rate-distortion theory heavily draws on Bayesian estimation and decision theory in order to model perceptual distortion and even aesthetic judgment.

Types

There are two basic lossy compression schemes:

- In *lossy transform codecs*, samples of picture or sound are taken, chopped into small segments, transformed into a new basis space, and quantized. The resulting quantized values are then entropy coded.

- In *lossy predictive codecs*, previous and/or subsequent decoded data is used to predict the current sound sample or image frame. The error between the predicted data and the real data, together with any extra information needed to reproduce the prediction, is then quantized and coded.

In some systems the two techniques are combined, with transform codecs being used to compress the error signals generated by the predictive stage.

Lossy versus lossless

The advantage of lossy methods over lossless methods is that in some cases a lossy method can produce a much smaller compressed file than any lossless method, while still meeting the requirements of the application.

Lossy methods are most often used for compressing sound, images or videos. This is because these types of data are intended for human interpretation where the mind can easily "fill in the blanks" or see past very minor errors or inconsistencies – ideally lossy compression is transparent (imperceptible), which can be verified via an ABX test.

Transparency

When a user acquires a lossily compressed file, (for example, to reduce download time) the retrieved file can be quite different from the original at the bit level while being indistinguishable to the human ear or eye for most practical purposes. Many compression methods focus on the idiosyncrasies of human physiology, taking into account, for instance, that the human eye can see only certain wavelengths of light. The psychoacoustic model describes how sound can be highly compressed without degrading

perceived quality. Flaws caused by lossy compression that are noticeable to the human eye or ear are known as compression artifacts.

Compression ratio

The compression ratio (that is, the size of the compressed file compared to that of the uncompressed file) of lossy video codecs is nearly always far superior to that of the audio and still-image equivalents.

- Video can be compressed immensely (e.g. 100:1) with little visible quality loss;
- Audio can often be compressed at 10:1 with imperceptible loss of quality;
- Still images are often lossily compressed at 10:1, as with audio, but the quality loss is more noticeable, especially on closer inspection.

Transcoding and editing

An important caveat about lossy compression is that converting (formally, transcoding) or editing lossily compressed files causes digital generation loss from the re-encoding. This can be avoided by only producing lossy files from (lossless) originals, and only editing (copies of) original files, such as images in raw image format instead of JPEG.

Editing of lossy files

Some editing of lossily compressed files without degradation of quality, by modifying the compressed data directly without decoding and re-encoding, is possible. Editing which reduces the file size as if it had been compressed to a greater degree, but without more loss than this, is sometimes also possible.

JPEG

The primary programs for lossless editing of JPEGs are `jpegtran`, and the derived `exiftran` (which also preserves Exif information), and Jpegcrop (which provides a Windows interface).

These allow the image to be

- cropped,
- rotated, flipped, and flopped, or
- converted to grayscale (by dropping the chrominance channel).

While unwanted information is destroyed, the quality of the remaining portion is unchanged.

JPEGjoin allows different JPEG images which have the same encoding to be joined without re-encoding.

Some changes can be made to the compression without re-encoding:

- optimize the compression (to reduce size without change to the decoded image),
- convert between progressive and non-progressive encoding,

The freeware Windows-only IrfanView has some lossless JPEG operations in its `JPG_TRANSFORM` plugin.

MP3

Splitting and joining

Mp3splt and Mp3wrap (or AlbumWrap) allow an MP3 file to be split into pieces or joined losslessly. These are analogous to `split` and `cat`.

Fission by Rogue Amoeba on the Macintosh platform will also allow you to join and split MP3 and m4a (Advanced Audio Coding) files without incurring an additional generational loss.

Gain

Various Replay Gain programs such as MP3gain allow the gain (overall volume) of MP3 files to be modified losslessly.

Metadata

Metadata, such as ID3 tags, Vorbis comments, or Exif information, can usually be modified or removed without modifying the underlying data.

Downsampling/compressed representation scalability

One may wish to downsample or otherwise decrease the resolution of the represented source signal and the quantity of data used for its compressed representation without re-encoding, as in bitrate peeling, but this functionality is not supported in all designs, as not all codecs encode data in a form that allows less important detail to simply be dropped.

Some well known designs that have this capability include JPEG 2000 for still images and H.264/MPEG-4 AVC based Scalable Video Coding for video. Such schemes have also been standardized for older designs as well, such as JPEG images with progressive encoding, and MPEG-2 and MPEG-4 Part 2 video, although those prior schemes had limited success in terms of adoption into real-world common usage.

Without this capacity, which is often the case in practice, to produce a representation with lower resolution or lower fidelity than a given one, one needs to start with the original source signal and encode, or start with a compressed representation and then decompress and re-encode it (transcoding), though the latter tends to cause digital generation loss.

Some audio formats feature a combination of a lossy format and a lossless correction which when combined reproduce the original signal; the correction can be stripped,

leaving a smaller, lossily compressed, file. Such formats include MPEG-4 SLS (Scalable to Lossless), WavPack, and OptimFROG DualStream.

Methods

Graphics

Image

- Cartesian Perceptual Compression: Also known as CPC
- DjVu
- Fractal compression
- HAM, hardware compression of color information used in Amiga computers
- ICER, used by the Mars Rovers: related to JPEG 2000 in its use of wavelets
- JPEG
- JPEG 2000, JPEG's successor format that uses wavelets, for Lossy or Lossless compression.
- JBIG2
- PGF, Progressive Graphics File (lossless or lossy compression)
- Wavelet compression
- S3TC texture compression for 3D computer graphics hardware

Video

- H.261
- H.263
- H.264
- MNG (supports JPEG sprites)
- Motion JPEG
- MPEG-1 Part 2
- MPEG-2 Part 2
- MPEG-4 Part 2 and Part 10 (AVC)
- Ogg Theora (noted for its lack of patent restrictions)
- Dirac
- Sorenson video codec
- VC-1

Audio

Music

- AAC
- ADPCM
- ATRAC
- Dolby AC-3

- MP2
- MP3
- Musepack (based on Musicam)
- Ogg Vorbis (noted for its lack of patent restrictions)
- WMA

Speech

- CELP
- G.711
- G.726
- Harmonic and Individual Lines and Noise (HILN)
- AMR (used by GSM cell carriers, such as T-Mobile)
- Speex (noted for its lack of patent restrictions)

Other data

Researchers have (semi-seriously) performed lossy compression on text by either using a thesaurus to substitute short words for long ones, or generative text techniques , although these sometimes fall into the related category of lossy data conversion.

Lowering resolution

A general kind of lossy compression is to lower the resolution of an image, as in image scaling, particularly decimation. One may also remove less "lower information" parts of an image, such as by seam carving.

Many media transforms, such as Gaussian blur, are, like lossy compression, irreversible: the original signal cannot be reconstructed from the transformed signal. However, in general these will have the same size as the original, and are not a form of compression.

Lowering resolution has practical uses, as the NASA New Horizons craft will transmit thumbnails of its encounter with Pluto-Charon before it sends the higher resolution images.