**Il Suono è un’energia** formata dalla vibrazione, tramite delle particelle d’aria che si scontrano generando onde. Come ogni forma di energia si disperde nell’ambiente in proporzione alla distanza La **pressione sonora**, detta anche pressione acustica, è rappresentata dalla variazione di pressione esercitata dal suono (Pascal) **RMS (Root Mean Square)** si tratta di un valore medio della pressione istantanea calcolato su un intervallo di tempo definito. **Decibel (SPL)** Può essere utile esprimere la pressione sonora in termini di decibel sonori quando si ha a che fare con problemi legati all'[udito](https://it.wikipedia.org/wiki/Udito), **Potenza Sonora** a differenza della pressione, è indipendente dalla distanza e, ad esempio, dalle dimensioni della stanza. È la potenza totale emessa dalla sorgente in tutte le direzioni. **Psicoacustica** studia la risposta psicologica e fisiologica associata al suono (rumore, voce, musica, ...). **Effetti psicoacustici** nella localizzazione suono, riconoscimenti suoni, effetti mascheramento (inganno tipo mp3) **Tipologie di suono** 🡪 periodici e aperiodici **Caratteristiche Onde sonore**

**Ampiezza :** determina intensità del suono

**Periodo(lunghezza d’onda (s)) e frequenza(Hz):** per i suoni periodici determinano l’altezza (pitch) del suono.

Frequenza misura le oscillazioni ed è proporzionale alla lunghezza d’onda. > lunghezza < frequenza

Orecchio umano tra i 20Hz e 20kHz. Tra i 3 e 4 Hz si sente meglio.

Frequenze acute si disperdono più facilmente rispetto alle frequenze basse.

**A-weighting (dBa)** adatta le misure alla differenze percezione dell’essere umano.

**Fase** differenza di distanza tra due medesimi punti. Somma di due onde uguali (una inversa) produce silenzio.

**Il timbro** determina il carattere/colore del suono. Il più complesso è il rumore bianco

**Spettro sonoro** (frequenze) modellato combinando tutte le onde presenti che ne determinano l’asse x e l’ampiezza y

**Inviluppo** 🡪 attacco, decadimento, sostegno, rilascio

**Spettrogramma** grafico in funzione di tempo, frequenza ed intensità

**Stereofonia** i medesimi contenuti sonori vengono riprodotti con differenti ampiezza (cuffietta sinistra suoni localizzati a destra più bassi)

**Audio analogico** immagazzina il segnale audio modificando la struttura fisica di una superficie. Un **segnale analogico** è un segnale continuo che varia nel tempo. Cambiando la pressione sonora nel microfono fa variare il voltaggio **dBV** e **dBu** sono unità di misura usate per le intensità di voltaggio. Come per i dB SPL si tratta di valori relativi a specifici valori di riferimento. **Range dinamico** è il rapporto tra ampiezza tra l’onda più forte possibile e il rumore di fondo (discorso 40 dB)

**Signal-to-noise(SNR)** differenza tra livello medio segnale e livello medio rumore di fondo-

**Livelli nominali,**sono livelli per cui gli apparecchi sono disegnati in modo da lavorare in maniera efficiente

**Headroom** è la differenza tra il livello nominale e livello massimo (distorsione)

**I VU meters** vengono utilizzati per rappresentare i livelli di intensità sonora negli apparecchi come i mixer.

**Un trasduttore (transducer)** è una device elettronico che converte energia da una forma all’altra. (mic o speaker)

**ADC (Analog to Digital Converter):** è un sistema che converte un segnale analogico in un segnale digitale. DAC è l’inverso

**Nella conversione analogico digitale**, viene utilizzato il processo di campionatura per acquisire, a intervalli regolari, dei campioni (samples) del segnale elettrico. **pulse code modulation (PCM):** tecnica di conversione permette di campionare a intervalli regolari l’ampiezza del segnale analogico.

**Dithering** algoritmo capace di spostare la percezione a 120 dB

**Decibels Full-Scale** È un unità di misura che mette in relazione l'ampiezza del segnale con l’ampiezza del picco massimo permesso da quel sistema (detto il full-scale). Negli apparecchi digitali, 0 VU ha normalmente un valore di −18 dBFS

**clipping**, una forma di distorsione molto aggressiva.

**Sample rate** è il numero di campioni per secondo (frequenza) acquisiti. Il massimo raggiungibile è la frequenza equivalente alla metà del sample rate (44.1 kHz freq max 22050 Hz). Questa frequenza è della frequenza di **Nyquist**

**Teorema di Nyquist-Shannon (sampling theorem)** Definisce la minima frequenza necessaria per [campionare](https://it.wikipedia.org/wiki/Campionamento_(teoria_dei_segnali)) un [segnale](https://it.wikipedia.org/wiki/Segnale_(fisica)) analogico senza perdere informazioni, e per poter quindi ricostruire il segnale analogico. Deve essere maggiore del doppio della sua frequenza massima

**Aliasing** Le frequenze oltre alla massima vengono riflesse all’interno dello spettro, generando frequenze inesistenti.

**Jittering** una lieve variazione nella precisione dei clock dei convertitori ADC o DAC. Questa lieve alterazione può modificare lo spettro delle fasi del segnale, riducendo conseguentemente la qualità di riproduzione.

**Formati audio**

* Non compressi (.wav, aiff)
* lossless ( .flac)
* Lossy (.mp3,.aac)

**Compressione Psicoacustica** lavora riducendo o approssimando la precisione di alcune componenti sonore considerate sotto la capacità di udito delle persone. (tipo .mp3)

**signal processing** per modificare in maniera intenzionale segnali audio.

**Tecniche di audioprocessing**

* equalizzazione e impiego di altri tipi di filtri
* modulazioni di ampiezza o di frequenza
* compressione, espansione e limiting
* aggiunta di delay e riverberi
* chorus, flanger, phaser
* pitch-shift e time-stretching
* effetti audio 3D

**Risposta impulsiva** è una funzione (del tempo) che coincide con l’output del sistema se sollecitato con un impulso. (delta di dirac)

**Risposta in frequenza** è una misura quantitativa dello spettro di output di un sistema, utilizzata per caratterizzare la dinamica del sistema. Una risposta in frequenza piatta, significa assenza di alterazioni sonore.

**Risposta in fase** simile a risposta in frequenza mette in relazione le differenze di fase fra il segnale di input e quello di output.

**La trasformata di fourier** scompone una funzione del tempo (quindi un segnale) nelle frequenze che la compongono. Permette quindi di passare dal dominio del tempo al dominio delle frequenze.

**Trasformata discreta di Fourier (DFT)** converte una sequenza finita di campioni di una funzione, in una sequenza (della stessa lunghezza) di campioni della trasformata di Fourier

**Linearità** significa che, se il sistema ha due o più inputs e due o più corrispondenti outputs ad ogni combinazione lineare degli inputs corrisponde una combinazione lineare degli outputs.

**Tramite segnali non lineari non è possibile utilizzare strumenti del signal processing**

**Linear time-invariant system (LTI)** significa che non è importante quando viene fatto partire il sistema. Per un determinato input, risulterà comunque sempre il medesimo output.

**Infinite Impulse Response (IIR)** Se batti le mani nel microfono all'ingresso, all'uscita avrai un suono che si estenderà (in teoria) all'infinito. Sono così perchè usano il feedback (quindi quello che esce in parte rientra, e quindi non è assurdo che duri all'infinito, sempre IN TEORIA). Più efficienti rispetto a FIR, facilmente parametrizzati in tempo reale  
**Finite impulse response (FIR)** è un filtro la cui risposta impulsiva (o risposta a qualsiasi input di lunghezza finita) è di durata finita, perché l’output va automaticamente a 0 in un tempo finito. NON usano feedback. Quindi, finito il clapclap, lasciano la coda (risposta all'impulso finita) e poi silenzio.

**Processori di dinamica** tipo compressori e limiter sono non lineari. Mantengono in memoria un certo numero di dati recenti che usano per adattare dinamicamente (l’inviluppo) del segnale d’uscita

**Immagine**

Una rappresentazione visiva del’ambiente che ci circonda bidimensionale o tridimensionale. **Luce** radiazione elettromagnetica (EMR). **EMR** classificata per lunghezza d’onda in radio, micro onde, infrarosso, luce visibile, ultravioletto, raggi X e raggi gamma **Luce visibile** viene emessa e assorbita in piccoli pacchetti chiamati fotoni, che esibiscono proprietà sia delle onde che delle particelle, questa caratteristica è la dualità onda-particella **Sistema** **visivo**: gli occhi processano la luce, il senso della vista comincia con la luce che attraversa la cornea e poi attraversa il cristallino, il risultato è un’immagine proiettata su un foglio di fotorecettori (retina) **colori** vengono generati dagli oggetti in funzione della loro capacità di riflettere le varie lunghezze d’onda presenti nella luce visibile. **Modello dei colori (RGB o CMYK)**

**Additiva**: somma dei colori derivati dai primari =bianco (somma colori modello RGB)

**Sottrattiva**: somma dei primari (partendo dai secondari) = nero (somma colori modello CMY)

**Isogrammi**: rappresentazione della distribuzione dei colori, per le immagini digitali rappresenta il numero di pixel in ogni range di colore (per le monocromatiche si usa istogramma di intensità)

**Fotografia** Una lente viene tipicamente utilizzata per la messa a fuoco della luce riflessa o emessa dagli oggetti, in modo da catturarla per un determinato periodo di esposizione, sulla superficie fotosensibile all’interno di una camera.

**Tipi di immagini** Raster(telecamere/foto) o vettoriali (costose)

**Sensori CCD** ha un unico amplificatore per tutti i pixels, utilizzati per camere digitali a basso costo

**CMOS** ogni pixel ha un suo amplificatore, maggior parte delle camere perché offre migliori prestazioni

**Sampling-rate:** determina risoluzione spaziale **Livello quantizzazione**: determina numero di tonalità di grigio o livelli di colore che vengono acquisiti

**Campionatura**: un’immagine fa parte del mondo reale quindi è analogica e la sua funzione è continua,la discretizza nello spazio trasformandola in una matrice di pixel **Quantizzazione**: si occupa della discretizzazione dell’intensità luminosa restituendo un valore nel caso delle tonalità di grigio, rispettivamente tre o più valori nel caso di un modello dei colori.(introduce rumore)

**Risol. Spaziale**: misura del più piccolo dettaglio visibile in un’immagine digitale, misurata in dots per inch (DPI)

**Risol. Intensità**: si riferisce al più piccolo dettaglio discernibile nel livello di intensità di grigio(8bit) o colore (16bit)

**Aliasing** Scalatura di un immagine, parti piccole hanno freq alta, soluzione filtri anti-aliasing **Dither** aggiunta di rumore per rimuovere il rumore di quantizzazione

**Interpolazione** è un'operazione spesso utilizzata per le immagini per eseguire zoom, rotazioni, e altre correzioni geometriche

**Formati** per raster sono .gif(lossy), .jpg (lossy), .tiff(tutti e due), .png(lossless) e .bmp. Contengono un header e descrizione immagine.

Le immagini digitali raster possono essere viste come matrici. Quindi su di esse è possibile eseguire sia operazioni di tipo array, sia matriciali. Anche nel caso dell’image processing si fa distinzione fra operazioni lineari e nonlineari.

**Nell’image processing la convoluzione** è l’operazione più importante perchè le immagini hanno l’informazione codificata nel dominio spaziale invece che in quello delle frequenze **La trasformata di Fourier** di un’immagine converte l’informazione chiara del dominio spaziale in una forma poco intellegibile nel dominio delle frequenze.

**Digital image processing** Trattamento dei colori dell’immagine, ricostruzione e ristrutturazione dell'immagine, segmentazione, operazioni di morfologia matematica, riconoscimento di forme e oggetti.

**Equalizzazione dell'istogramma l**'obiettivo è di ridistribuire le intensità nell'istogramma, fornendo, come conseguenza, maggiore contrasto a zone che hanno basso contrasto locale **Normalizzazione dell’immagine** vengono modificate le intensità dei pixels (estendendone il range) per ottenere una migliore occupazione del range dinamico disponibile.

**Gaussian blur** Viene spesso utilizzata(tramite convoluzione) per ridurre il rumore e la presenza di dettagli nelle immagini.

**Box blur** ogni pixel è la media dei suoi vicini nell’immagine di partenza. I box blur vengono utilizzati per approssimare un Gaussian blur. Specificati tramite matrice 3X3.

**Sharpening** Un’immagine è a fuoco se ha buona risoluzione e un buon contrasto sui contorni (edges). Più l’immagine ha risoluzione, meglio può essere messa a fuoco. Per quanto concerne il contrasto dei bordi, si tratta di un fattore soggettivo. Può essere migliorato applicando filtri che migliorano il contrasto dei contorni.

**Convoluzione** Nel trattamento d’immagine, la convoluzione è il processo di aggiungere ogni elemento ai propri vicini, moltiplicato per i valori specificati in un kernel di dimensione 3x3.

**Kernel** Possono essere utilizzati per implementare una moltitudine di filtri, incluse alcune soluzioni per la detezione dei contorni.

**Normalizzazione del kernel** somma di tutti gli elementi della matrice fa 1, Questa soluzione garantisce che che i pixels siano in media luminosi nell’immagine risultante quanto lo sono nell’immagine originale

**Detezione dei contorni (edge detection)** ha lo scopo di marcare i punti in cui l'intensità luminosa cambia bruscamente. (eliminando i dettagli non rilevanti)

**Trasformata di Fourier 2D** immagini dominio dello spazio invece che in quello del tempo. In maniera simile alla convoluzione, La DFT 2D viene utilizzata per applicare varie tipologie di filtri.

**Denoise** rumore dovuto alle meccaniche interne al prodotto utilizzato (richiesto rimozione ruomore)

**Rumore Salt-and-pepper** occorrenze sparse di pixels neri o bianchi. Un metodo efficace per la sua riduzione è l’utilizzo di un filtro mediano.

**Rumore gaussiano** rumore bianco, in cui i valori sono tutti indipendenti e distribuiti in maniera identica. Una delle tecniche convenzionali per la rimozione del rumore gaussiano è quella della media tramite convoluzione

**Restoration: Wiener filter** I filtri più tradizionali eseguono la separazione tra segnale e rumore a condizione che questi occupino diverse bande di frequenza. Si assume di avere conoscenza delle caratteristiche spettrali del segnale originale e del rumore, e si cerca il filtro LTI il cui risultato sia "il più vicina possibile" al segnale originale (minimizzando una prefissata misura dell'errore). Questo filtro è spesso utilizzato nel processo di deconvoluzione.

Segmentazione suddivisione in più segmenti l’immagine L'obiettivo è quello di semplificare/modificare la rappresentazione dell’immagine in qualcosa che sia più facile e sensato da analizzare. utilizzata per identificare forme e contorni.

**Thresholding** conversione di un immagine tramite una soglia (bianco o nero), se basato su instrogramma è più efficace perché passa una sola volta. Picchi e valli dell’istogramma vengono utilizzati per identificare i clusters all’interno dell’immagine. Può funzionare sia sulla base delle intensità di grigio, sia sulla base dei colori.

**K-means** L’algoritmo di K-means sfrutta una tecnica iterativa per partizionare l’immagine in K clusters.

**La morfologia matematica** sta alla base delle tecniche di morphological image processing, che consistono nell'impiego di alcuni operatori per trasformare le immagini (solitamente di tipo binario, ma si può applicare anche ad immagini in scale di grigio). I quattro operatori principali sono: erosioni, dilatazione, apertura e chiusura. L’idea di base della morfologia binaria è di testare l'immagine con una struttura semplice predefinita, per verificare se questa struttura è compatibile o incompatibile con le forme presenti nell’immagine. La struttura di test viene chiamata structuring element

**Erosione** di un quadrato di dimensione 10 con un disco di raggio 2 è un quadrato di dimensione 6

**Dilatazione** di un quadrato di dimensione 10 con un disco di raggio 2 è un quadrato di dimensione 14 con gli angoli arrotondati.

**Cuda**

**Ostacoli tecnici Memory wall:** crescente divario tra la velocità di processore e memoria. Per mascherare la latenza e quindi evitare il collo di bottiglia le dimensioni e i livelli delle cache crescono sempre di più. **ILP wall (Instruction Level Parallelism**) aumentando il numero di core dei processori cresce la difficoltà nel programmarli, l’ideale sarebbe occupare in modo equivalente tutti i core del processore. **Power wall:** aumentando la potenza aumenta anche il consumo di energia e di conseguenza la produzione di calore. Questo è il motivo per il quale i processori tendono ad avere tutti una frequenza “bassa”.

**Accelerazione hardware** consiste nell'uso di hardware appositamente disegnato per eseguire funzioni specifiche più velocemente di quanto sia possibile con software su una CPU generica Difatti gli acceleratori hardware sono progettati per il calcolo compiti intensivi, quindi l’esempio tipico sono le GPU

**GPU** sono processori specializzati con memoria dedicata e con un design multi-core ottimizzato per SIMD e FPU (spesso richieste per la grafica di rendering). Le GPU dedicano proporzionalmente più transistor alle unità aritmetiche / logiche e meno alla cache e al controllo di flusso in confronto a una CPU (infatti non vi è branch prediction, speculative execution, ecc …). Le GPU hanno in genere anche una memoria DRAM più veloce (in termini di bandwith) rispetto alle CPU. Le GPU hanno un'architettura di throughput parallela per ottimizzare l'esecuzione di molti thread simultanei lentamente, piuttosto che eseguire un singolo thread molto rapidamente **Nascondere Latenza** Anche con una larghezza di banda elevata, la memoria rimane molto lenta rispetto ai core Soluzione molti thread rispetto core, overhead context switch moto bassi, thread non vengono swappati ma rimangono sullo stesso core.

**Programmare GPU** Di norma si usano le librerie per programmare in quanto l’uso di quest’ultime consente di programmare senza una conoscenza approfondita della sua architetturaIl **Compute Unified Device Architecture (CUDA)** consente di sfruttare la potenza delle GPU Nvidia.

**kernel** una funzione che esegue sulla GPU richiamata dall’host. Quidni l’esecuzione su GPU avviene tramite i kernel che sono programmati sfruttando CUDA. Ogni kernel deve eseguire il minor lavoro possibile

**Memoria unificata** è un sistema di gestione della memoria fornendo un unico spazio accessibile da tutte le GPU e CPU nel sistema. La migrazione delle pagine tra memoria host e memoria device (che avviene automaticamente) consente alla CPU di accedere alla cache L2 della GPU la quale serve a ridurre la latenza della memoria.

**Esecuzione del kernel** Quando un programma CUDA sulla CPU richiama un kernel, i blocchi della griglia vengono distribuiti a tutti i core della GPU disponibili. Ogni blocco di thread esegue parallelamente su un multiprocessore e più blocchi di thread possono essere eseguiti in modo concorrente sullo stesso multiprocessore. (numero thread per blocco limitato, consente singola chiamata di un kernel a più thread, blocchi hanno id e possono essere mono o bidimensionali). Il multiprocessore crea, gestisce, schedula ed esegue i thread in gruppi da 32 chiamati **wraps**. Ogni singolo thread che costituisce un wrap parte dallo stesso punto nel programma. Se durante l’esecuzione di un wrap è richiesto a più thread di eseguire un’operazione atomica sulla stessa area di memoria questa operazione verrà eseguita in modo seriale in ordine casuale.

**\_device\_** Eseguibile e richiamabile solo dal device, risiede nella global memory space è liftime of an application ed è accessibile da tutti i threads all’interno della griglia e dall’host attraverso la runtime library **\_\_global\_\_** Eseguibile sul device, richiamabile solo dall’host **\_\_costant\_\_** Risiede nella (global) costant memory space, è lifetime of an app. ed è accessibile da tutti i threads all’interno della griglia e dall’host **\_\_host\_\_** Eseguibile e richiamabile solo da host **\_\_shared\_\_** Risiede nella shared memory space del thread block. Ha lifetime sul blocco. È solo accessibile da tutti i threads attraverso i blocchi

**Register Memory** la forma più veloce di memoria sulla GPU, risiede sul multi-core, è unicamente accessibile dai threads ed è lifetime of the thread. **Shared Memory** Può essere veloce come un registro quando non ci sono conflitti sui banchi di memoria o quando si legge dallo stesso indirizzo. Accessibile da qualsiasi thread del blocco da cui è stato creato. È lifetime of the block. **Global Memory** Potenzialmente è 150x più lenta di un registro o della shared memory. Bisogna fare attenzione alla scrittura e lettura quando non sono coerenti tra di loro. **Local Memory** velocità(global) accessibilità (register) usato automaticamente da CUDA per: register overflow, array addressed by parameters in register memory e letture di memoria globale. Costant Memory Cached access to global memory, di sola lettura condiviso da tutti i threads e dall’istruction cache **Texture Memory** Cached access to global memory, di sola lettura. Ottimizzata per accessi 2D

**Syncthreads** I thread all'interno di un blocco possono cooperare condividendo i dati attraverso una memoria condivisa e sincronizzando la loro esecuzione per coordinare gli accessi alla memoria. I punti di sincronizzazione sono specificati nel kernel attraverso la chiamata \_\_syncthreads() (eseguito attraverso una barriera in cui tutti i thread nel blocco devono attendere prima di poter procedere) Per una cooperazione efficiente, la memoria condivisa dovrebbe essere una low-latency memory vicino a ciascun core del processore.

\_\_global\_\_ void MatMulKernel(Matrix A, Matrix B, Matrix C) {

// Each thread computes one element of C

float Cvalue = 0;

int row = blockIdx.y \* blockDim.y + threadIdx.y;

int col = blockIdx.x \* blockDim.x + threadIdx.x;

for (int e = 0; e < A.width; ++e)

Cvalue += A.elements[row \* A.width + e] \* B.elements[e \* B.width + col];

C.elements[row \* C.width + col] = Cvalue;

}

typedef struct {int width;int height;float\* elements;} Matrix;

#define BLOCK\_SIZE 16

void MatMul(const Matrix A, const Matrix B, Matrix C) {

Matrix d\_A;

d\_A.width = A.width; d\_A.height = A.height;

size\_t size = A.width \* A.height \* sizeof(float);

cudaMalloc(&d\_A.elements, size);

cudaMemcpy(d\_A.elements, A.elements, size, cudaMemcpyHostToDevice);

Matrix d\_B;

d\_B.width = B.width; d\_B.height = B.height;

size = B.width \* B.height \* sizeof(float);

cudaMalloc(&d\_B.elements, size);

cudaMemcpy(d\_B.elements, B.elements, size, cudaMemcpyHostToDevice);

// Allocate C in device memory

Matrix d\_C;

d\_C.width = C.width; d\_C.height = C.height;

size = C.width \* C.height \* sizeof(float);

cudaMalloc(&d\_C.elements, size);

dim3 dimBlock(BLOCK\_SIZE, BLOCK\_SIZE);

dim3 dimGrid(B.width / dimBlock.x, A.height / dimBlock.y);

MatMulKernel<<<dimGrid, dimBlock>>>(d\_A, d\_B, d\_C);

/ Read C from device memory

cudaMemcpy(C.elements, Cd.elements, size, cudaMemcpyDeviceToHost);

// Free device memory

cudaFree(d\_A.elements);

cudaFree(d\_B.elements);

cudaFree(d\_C.elements);}

**Video Processing e Computer Vision**

**Videocamere** vengono registrate sequenze di immagini. **Riproduzion**e fenomeno **beta-phi**, il cervello ricostruisce un'esperienza di movimento apparente (10/12 frame) **Supporti per il salvataggio** Betamax, cassette VHS, DVD e dischi Blu-ray

**Tecnologie per display**

**CRT (Cathode Ray Tube)**: è un tubo sottovuoto contenente uno o più emettitori di elettroni combinato con uno schermo fluorescente. Per creare le immagini accelera e deflette i fasci di elettroni sullo schermo. **DLP (Digital Light Processing)**: è una tecnologia per video proiettori che sfrutta micro-specchi digitali. Viene normalmente impiegata per i proiettori utilizzati negli ambiti commerciali o dell’insegnamento. **PDP (Plasma Display Panel):** è un pannello piatto che utilizza piccole celle contenenti gas ionizzato caricato elettricamente. Utilizzato principalmente per grossi schermi. **LCD (Liquid Crystal Display):** è un pannello piatto che utilizza cristalli liquidi modulati dalla luce. Per produrre la luce vengono utilizzati dei riflettori o delle backlight (tipicamente delle luci fluorescenti nelle LCD-TV o dei LED nelle LED-TV). **OLED (Organic Light-Emitting Diode):** è una tecnologia che per emettere la luce sfrutta LED, che come elemento elettro-luminescente hanno uno strato di materiale organico capace di produrre differenti tipi di luce in risposta alla sollecitazione elettrica ricevuta.

**Risoluzioni** SD(Standard Definition): 576i×360 HD(High Definition): 1280×720 UHD(Ultra High Definition): 3840×2160

**Aspect Ratio** rapporto larghezza e altezza.(4:3)

**Video Processing** (modifica dell’aspect ratio; zoom digitale; aggiustamento di brightness, contrast, hue, saturation, sharpness, gamma; conversione del frame rate; aggiustamento del colore; riduzione del rumore; tecniche di upscaling)

**intra-frame** tecniche attuate a livello di singola immagine

**inter-frame** tecniche che sfruttano l’informazione temporale esistente fra più di una immagine

**Formati container per il multimedia (AVI, MP4, FLV)**

**Inter-frame prediction** compressione, in particolare avvantaggiandosi della correlazione tra immagini vicine nella sequenza (H.264) **Motion JPEG** è un formato di compressione video intraframe in cui ogni frame è compresso separatamente come immagine JPEG.

**Computer Vision** comprende tecnologie e metodi per acquisire, modificare ed analizzare singole immagini o sequenze di immagini digitali, con lo scopo ultimo di estrapolare informazioni

**Struttura** Acquisizione imm., Pre-processing, estrazione feature, detection, processing (verifica, classificazione)

**Background substraction** ha l’obbiettivo di estrarre elementi in primo piano all’interno di un video (identificare l’oggetto in movimento tra frame diff) **tecniche Frame differencing, mean filter, gaussian average**

**Frame Differencing** sottrazione d’immagine per ogni pixel intensità luminosa solo per i pixels che hanno subito modifiche

**Video motion traking** Lo scopo delle tecniche di video tracking è di associare gli oggetti in frames video consecutivi rappresentazione e localizzazione del target usate tecniche di kernel-based tracking, come il mean shift, o di contour tracking. filtraggio e associazione dei dati applicati Kalman filter o Particle filter.

**Mean shift** basandosi sugli istogrammi dei colori dell’oggetto presente nell’immagine precedente. Utilizza l’algoritmo di mean shift per identificare il picco in prossimità della vecchia posizione dell’oggetto.

**Audio e video streaming**

Oggi la trasmissione di contenuti multimediali avviene tramite soluzioni di streaming audio/video che sfruttano Internet come canale di comunicazione.

**Principali mezzi trasmissivi (uno a uno)**

**Linee telefoniche** permette il collegamento di un telefono alla rete telefonica. Ogni linea di norma è associata ad un numero telefonico (oggi DSL)

**Trasmissioni radio** tramite onde radio per trasmettere informazione. La trasmissione modulando sistematicamente le proprietà di onde elettromagnetiche trasmesse nello spazio (Ampiezza,frequenza, fase, lunghezza impulso)

**Broadcasting** distribuzione di contenuti audio/video tramite mezzo di comunicazione di massa (uno a molti)

**Radio Broadcasting** oggi può passare anche via cavo o più genericamente in Internet o AM/FM

**AM: (Amplitude Modulation**) in corrispondenza dell’informazione da trasmettere viene modulata l’ampiezza in base all’intensità del segnale.

**FM: (Frequency Mosulation)** modulazione di frequenza nella banda. soluzione ai problemi di AM che produceva rumore nella ricezione (RFI = Radio Frequency Interferation).

**Televisione via cavo** Si utilizzano cavi coassiali al posto delle onde radio

**Televisione via satellite** Il segnale viene ricevuto da un’antenna

**Digitale terrestre** Per la radio viene utilizzato il sistema **DAB (Digital Audio Broadcasting)**. Il segnale analogico viene digitalizzato e compresso con formati come l’MP2, poi trasmesso con scemi di modulazione digitale Per la televisione digitale **terrestre (DTT o DTTV)** sfrutta le onde radio terrestri per trasmettere segnali TV digitali

**Radio e televisione internet** soluzione VOIP (voice over ip), ROIP (radio over ip), IPTV (internet protocol television)

**Streaming** audio/video trasmessi tramite internet, non necessario scaricare il contenuto, riproduzione istantanea

**Live media** permette accesso ad un evento live (twitch) **Time-shifted media** accesso programmazione passata **Video on demand (VOD)** accesso ad un catalogo (Netflix)

**Live Streaming** necessità strumentazione, encoder real-time per digitalizzazione, strumento di publishing e **content delivery network (CDN)** per distribuirli

**Problemi** i problemi possono essere di **packet-loss, packet-delay, network jitter (**causata dalle code interne ai router congestionati)

**Unicast** inviano una copia separata dello stream server a ogni utente connesso **Multicast** Viene spedita un’unica stream dalla sorgente agli utenti. (non sempre disp)

**CDN** responsabile dell’eventuale transcoding e della distribuzione dei contenuti multimediali agli utenti

**Ravenna**: è una tecnologia di streaming audio over IP per LAN (RTP/RTCP e SDP o RTSP)

**Lo streaming su WAN** è più complesso a causa della presenza di routers, firewalls e dei differenti ISP che riducono la possibilità di connessione

**Real-Time Transport Protocol**, livello applicazione per servizi di comunicazione audio/video in real-time over IP (telefonia, teleconnferenza, televisione e streaming internet). Viene usato con **UDP** e in combinazione con RTP Control Protocol (RTCP) per mantenere sincronizzazione delle streams. RTP supporta anche multicast

**MPEG-DASH** contenuti multimediali vengono spezzati in una sequenza di segmenti (ad esempio di 10 secondi) messi a disposizione in una varietà di bitrates differenti. In base alle condizione di rete, il dispositivo di ricezione è responsabile della scelta del formato adeguato da scaricare e riprodure

**GStream**

È un framework opensource che permette la creazioni di applicazioni multimediali. **Un elemento** è un componente della pipeline il quale ha delle proprietà **Necessità di 3 elementi** di base: source, filter e sink Pipelines insiemi di elementi collegati sequenzialmente. **Pipeline di base** -> almeno una source, almeno un sink e uno o più elementi filter descritta da una serie di caratteri separati da !

*gst-launch-1.0 audiotestsrc wave=0 freq=440 ! audioconvert ! autoaudiosink*

*gst-launch-1.0 videotestsrc ! videoconvert ! autovideosink*

*gst-launch-1.0 videotestsrc pattern=1 ! videoconvert ! autovideosink*

*gst-launch-1.0 playbin uri=https://download.blender.org/peach/bigbuckbunny\_movies/BigBuckBunny\_320x180.mp4*

*gst-launch-1.0 filesrc location=BigBuckBunny\_320x180.mp4 ! decodebin ! autovideosink*

*gst-launch-1.0 filesrc location=BigBuckBunny\_320x180.mp4 ! decodebin name=pippo pippo.src\_0 ! autovideosink pippo.src\_1 ! autoaudiosink*

*gst-launch-1.0 filesrc location=BigBuckBunny\_320x180.mp4 ! decodebin ! openh264enc ! openh264dec ! videoconvert ! autovideosink*

*gst-launch-1.0 filesrc location=BigBuckBunny\_320x180.mp4 ! decodebin ! openh264enc ! rtph264pay ! udpsink host=127.0.0.1 port=9001*

*gst-launch-1.0 udpsrc port=9001 caps = "application/x-rtp, media=(string)video, clock-rate=(int)90000, encoding-name=(string)H264, payload=(int)96" ! decodebin ! videoconvert ! autovideosink*

**Machine learning**

Il **machine learning** consiste nella realizzazione di algoritmi che imparano dai dati in modo da poter fare successivamente previsioni su di essi. In varie situazioni gli algoritmi si rivelano più efficaci di programmi scritti a mano, perché producono autonomamente un modello a partire da dati campione. I tipi sono:

**Supervised learning** si educa l’algoritmo tramite degli input e degli output in questo modo internamente trova le regole.L’apprendimento può essere fatto in esecuzione tramite feedback (**active learning**)

**Reinforcement learning** unica forma di feedback l’output viene approvato o respinto

**Unsupervised learning** nessuna informazione viene fornita all’algoritmo, che è costretto a scoprire in maniera autonoma la struttura dei dati in input.

**Reti Artificiali** sono utilizzate come strumenti non lineari per la modellazione di programmi complessi. Sono più semplici da allenare rispetto a machine learning ma hanno una natura di black-box (non si sa il contenuto) è necessario allenarlo con tanti dati.

**Nel deep learning,** ogni layer impara a trasformare i dati di input in una rappresentazione ogni volta leggermente più astratta e composta. È capace di migiorare le performances all’aumentare dei dati di apprendimento.(usato spesso nel multimedia)

int main(int argc, char \*\*argv){

namedWindow("image", WINDOW\_AUTOSIZE);

Mat image;Mat grayImage;Mat threshImage;

Mat filterImage;Mat borderImage;

int typeFilter = 0;int dim = 5;

image = imread("path/to/image/lego.png", CV\_LOAD\_IMAGE\_COLOR);

cvtColor(image, grayImage, cv::COLOR\_BGR2GRAY);

cin >> typeFilter;

cin >> dim;

switch (typeFilter) {

case 0:cv::blur(grayImage, filterImage, Size(dim, dim), Point(-1, -1), BORDER\_DEFAULT);break;

case 1:cv::GaussianBlur(grayImage, filterImage, Size(dim, dim), BORDER\_DEFAULT);break;}

threshold(filterImage, threshImage, 234, 255, THRESH\_BINARY\_INV);

std::vector<std::vector<cv::Point> > contours;

cv::findContours(threshImage, contours, CV\_RETR\_LIST, CV\_CHAIN\_APPROX\_SIMPLE);

cv::drawContours(image, contours, -1, CV\_RGB(0, 0, 0), 2.0);

String label;cv::RotatedRect rotatedRectangle;

for (auto& point : contours) {

rotatedRectangle = minAreaRect(point);

label = getValue(rotatedRectangle.size.width) + "x" + getValue(rotatedRectangle.size.height);

rotatedRectangle.center;

int x = rotatedRectangle.center.x;int y = rotatedRectangle.center.y;

Vec3b colors = image.at<Vec3b>(y, x);

point[0].y -= 5;

putText(image, label + getColor(colors), point[0], FONT\_HERSHEY\_PLAIN, 1.0, CV\_RGB(0, 255, 0), 2.0);}

cv::imshow("image", image);cv::waitKey(0);return 0;

}

std::vector<Tone> tones;

void printText(std::string numbers) {

int repetitions = -1;

if (numbers.at(0) == '\n') numbers.replace(0, 1, "");

numbers.push\_back('\n');

for (int i = 0; i < numbers.length(); i++) {

if (numbers.at(i) == '\n') {

int position = numbers.at(i - 1) - '0';

if (position > 0) position--;

std::cout << tones.at(position).decode(repetitions);

repetitions = -1;}else {repetitions++;}}}

int main(int argc, char \*\*argv){

File input = File::getCurrentWorkingDirectory().getChildFile("SignalToAnalyse.wav");

if (input.exists() == false) {

std::cerr << "File doesn't exists." << std::endl;

return 1;}

AudioFormatManager fmgr;

fmgr.registerBasicFormats();

ScopedPointer<AudioFormatReaderSource> source = new AudioFormatReaderSource(fmgr.createReaderFor(input), true);

unsigned int numChannels = source->getAudioFormatReader()->numChannels;

double sampleRate = source->getAudioFormatReader()->sampleRate;

std::cout << "Channels=" << numChannels << " sampleRate=" << (int)sampleRate << std::endl;

AudioBuffer<float> myBuffer(numChannels, sampleRate / 10);

AudioSourceChannelInfo info(myBuffer);

source->prepareToPlay(512, sampleRate);

// Variabili

const int FFTsize{ 1024 };

float buffer[FFTsize \* 2];

CircularBuffer<float> circularBuffer{ FFTsize, 64 };

MyFilter<float> filter{ 690,1650,sampleRate };

int ripetizioni[13];

std::string s;

tones.push\_back(Tone(697.f, 1209.f, '1', ""));

tones.push\_back(Tone(697.f, 1336.f, '2', "ABC"));

tones.push\_back(Tone(697.f, 1477.f, '3', "DEF"));

tones.push\_back(Tone(770.f, 1209.f, '4', "GHI"));

tones.push\_back(Tone(770.f, 1336.f, '5', "JKL"));

tones.push\_back(Tone(770.f, 1477.f, '6', "MNO"));

tones.push\_back(Tone(852.f, 1209.f, '7', "PQRS"));

tones.push\_back(Tone(852.f, 1336.f, '8', "TUV"));

tones.push\_back(Tone(852.f, 1477.f, '9', "WXYZ"));

tones.push\_back(Tone(941.f, 1209.f, '\*', ""));

tones.push\_back(Tone(941.f, 1336.f, '0', " "));

tones.push\_back(Tone(941.f, 1477.f, '#', ""));

while (source->getNextReadPosition() < source->getTotalLength()){

source->getNextAudioBlock(info);

const int numSamples = info.buffer->getNumSamples();

const int numChannels = info.buffer->getNumChannels();

const float \*\*data = info.buffer->getArrayOfReadPointers();

for (int channel = 0; channel < numChannels; channel++){

const float \*channelData = data[channel];

for (int sampleIndex = 0; sampleIndex < numSamples; sampleIndex++) {

if (circularBuffer.process(channelData[sampleIndex])) {

circularBuffer.copyData(buffer);

filter.process(buffer, FFTsize);

dsp::FFT fft{ 10 };

fft.performFrequencyOnlyForwardTransform(buffer);

buffer[0] = 20.0f \* log10f(buffer[0] / (float)FFTsize / 2.0f);

for (int i = 1; i < FFTsize / 2; i++) {

buffer[i] = 20.0f \* log10f((2.0f \* buffer[i]) / (float)FFTsize / 2.0f);}

bool found = false;

for (int i = 0; i < 12; i++) {

Tone t = tones.at(i);

if (t.lookup(buffer, FFTsize / 2)) {

found = true;

if (ripetizioni[12] > 120) {

s.push\_back('\n');

for (int j = 0; j < 12; j++)

ripetizioni[j] = 0;}

ripetizioni[12] = 0;

ripetizioni[i]++;

if (ripetizioni[i] > 12) {

s.push\_back(t.getValore());

ripetizioni[i] = 0;}}}

if (!found) {

ripetizioni[12]++;}}}}}

printText(s);return 0;}}

// 1. convert in grayscale cvtColor(frame, grayScaleFrame, CV\_BGR2GRAY);// 2. use the background subtraction MOG2 algorithm to obatin the fore ground image pMOG2->apply(grayScaleFrame, fgMaskMOG2);

// 3. filter the noise by using one or more morphological filters Mat kernel = cv::getStructuringElement(cv::MORPH\_RECT, cv::Size(2, 2));cv:erode(fgMaskMOG2, filterFrame, kernel);cv::dilate(filterFrame, filterFrame, kernel); cv::GaussianBlur (filterFrame, filterFrame, Size(), 2);apply a threshold to better define the foreground

cv::threshold(filterFrame, thresholdFrame, 50, 255, cv::THRESH\_BINARY);// 5. if needed, use one ore more morphological filters to improve the fore ground definition// 6. find a way to track the cars using the camshift algorithm (hint: implement an object to handle the tracker)Rect copyTrackWindowLeft = trackWindowLeft;Rect copyTrackWindowRight = trackWindowRight;RotatedRect TrackBoxLeft = CamShift(thresholdFrame, copyTrackWindowLeft,

TermCriteria(TermCriteria::EPS | TermCriteria::COUNT, 10, 1));

RotatedRect TrackBoxRight = CamShift(thresholdFrame, copyTrackWindowRight,

TermCriteria(TermCriteria::EPS | TermCriteria::COUNT, 10, 1));

// 7. extract the requested information (number of vehicles, number of vehicles for each direction, timestamp, color)

// 8. save ONE Nj for the vehicle with the requested information written on

if (TrackBoxLeft.size.width > 0 && xLeft <= TrackBoxLeft.center.x) {cntLeft = 0;

trackWindowLeft = TrackBoxLeft.boundingRect();

xLeft = TrackBoxLeft.center.x;if (TrackBoxLeft.center.x > 300) {// Obtaining color to parse

Mat frameAuto = frame(trackWindowLeft);

Mat filteredAuto = filterFrame(trackWindowLeft);

Scalar color = mean(frameAuto, filteredAuto);

// Construction of a color class

Color autoColor{ int(color[0]),int(color[1]),int(color[2])};

// Cloning frame to save image

Mat clone = frame.clone();// Regnognization of a color and construction of image's label ostringstream str;

str << "Frame: " << frameNumber << " , Color: " << autoColor.label() << " from left to right";

putText(clone, str.str(), Point(10, 30), FONT\_ITALIC, 0.4, cvScalar(0, 0, 255));

string name = "./fromLeft" + to\_string(cntAutoLeft) + ".jpg";

imwrite(name, clone);

// Setting starter parameters

trackWindowLeft = Rect(60, 170, 100, 100);

xLeft = 70;

// Incrementation of the auto detected variable

cntAutoLeft++;}} //rifare al posto di left🡪 right