Multimedijske tehnologije

Prof.dr.sc. Mario Kovač

Što ćemo obraditi...

- U prethodna dva bloka predavanja dobili ste osnovni pregled multimedijskih tehnologija
- U ovom zadnjem bloku cilj će nam biti vidjeti kako se neki od algoritama implementiraju u stvarnom svijetu : IZVEDBENI POGLED NA TEHNOLOGIJE
- S obzirom da je detaljna analiza i pregled algoritama koje ćemo spominjati izvan predviđenog opsega ovog predmeta i obrađuje se na diplomskom studiju mi ćemo dati osnovne informacije potrebne za razumijevanje...
- Ovo je i cilj preddiplomskog studija....

Koji podaci...

- Analize i primjeri biti će rađeni na slikovnim i video podacima jer su oni i najzahtjevniji
- Pretpostavlja se da ste dobro naučili prethodno gradivo
- Bez obzira na "moćne" računalne sustave u uporabi danas količine podataka koje sadrže slike i video zahtjevaju pažnju pri projektiranju

Količine podataka

- Jednostavan izračun daje nam okvirne količine nekih tipičnih formata koje ste upoznali

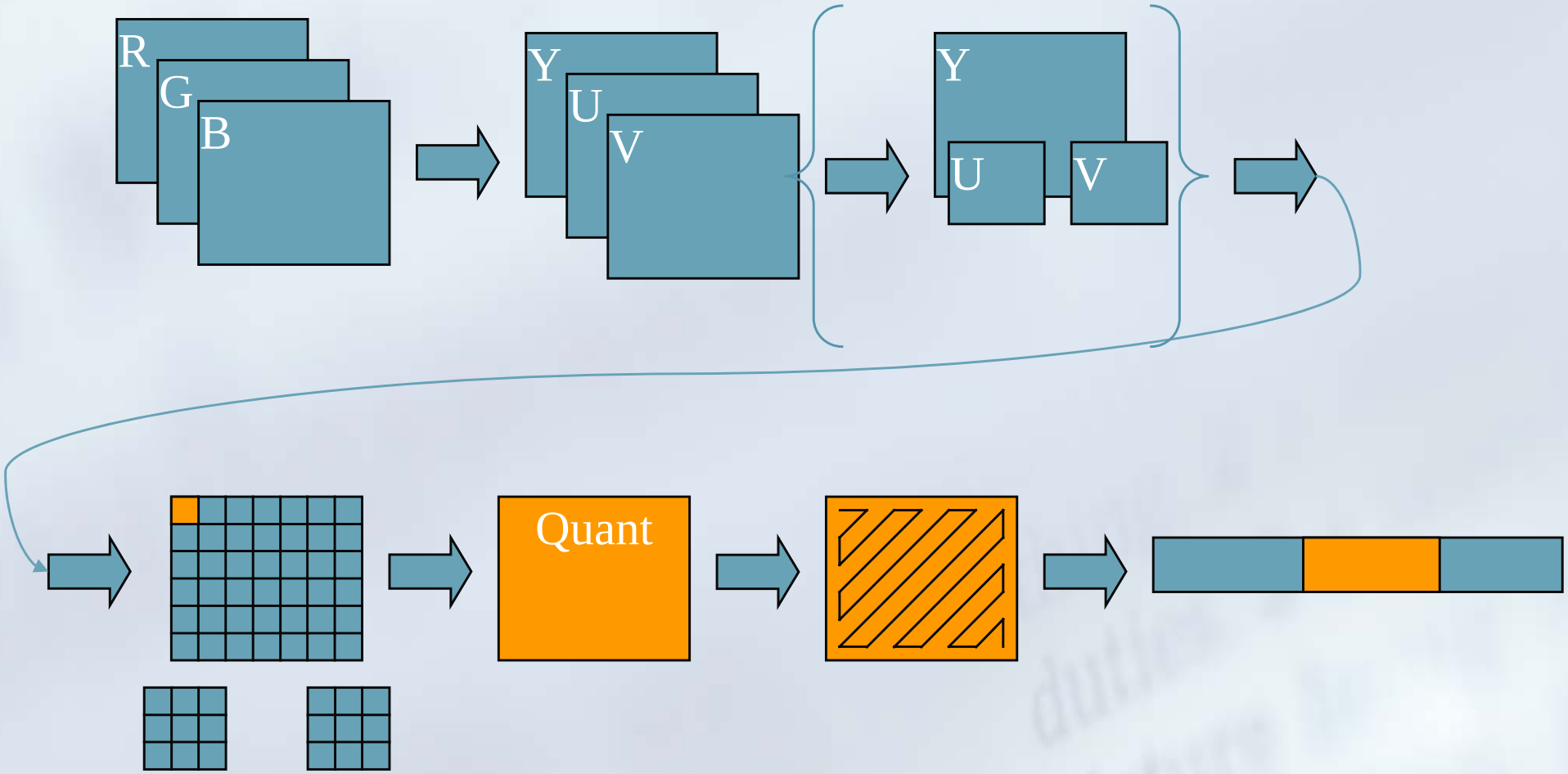
Rezolucija slike	**Bita/pikselu**	**Veličina**
640x480	8	307kB
1280x1024	24	3,9MB

Video rezolucija	**Slika/s**	**Veličina/s**
640x480 (8bpp)	10	3 MB/s
1280x1024 (24bpp)	30	120 MB/s

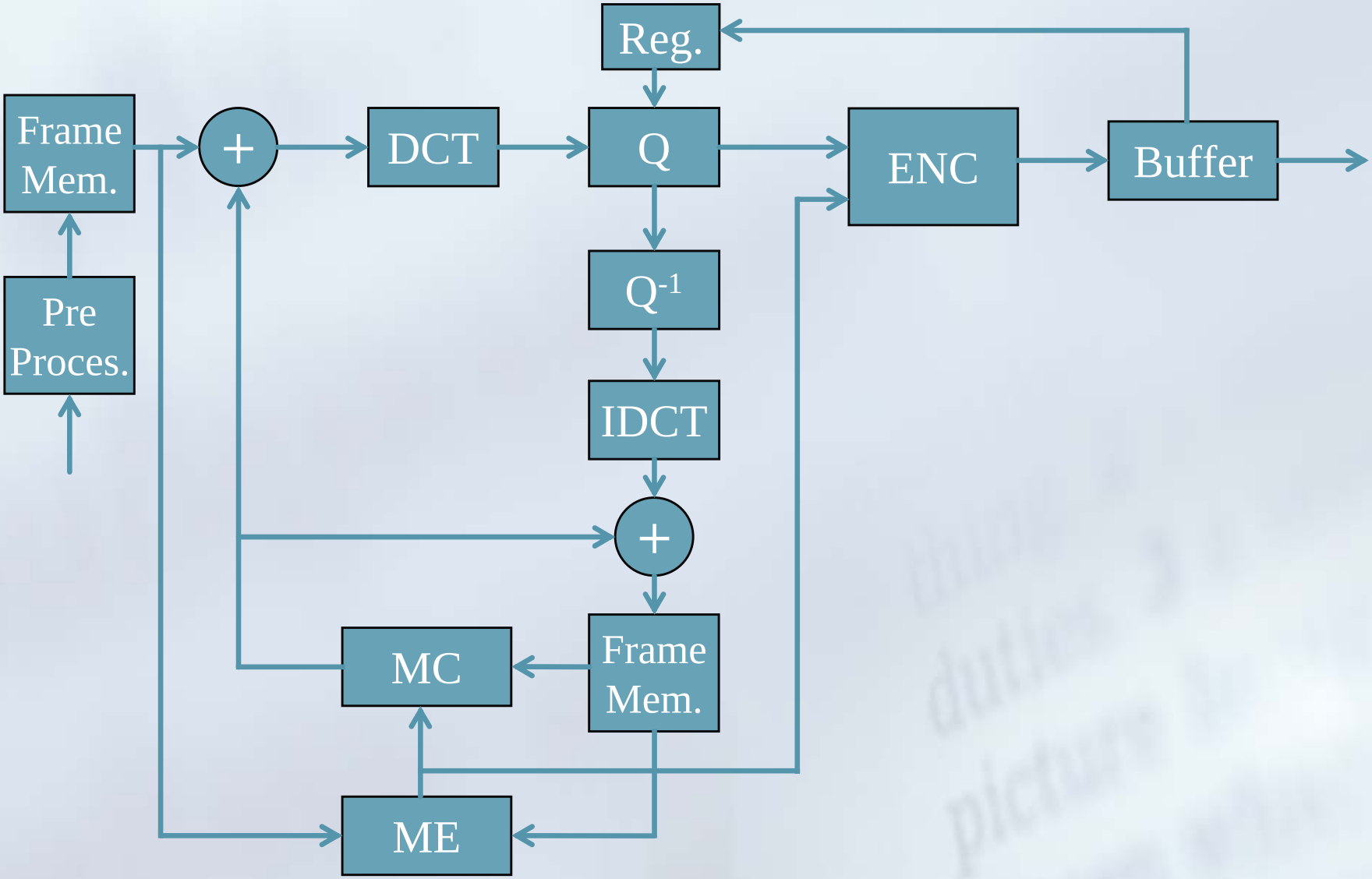
Najzahtjevnije operacije

- Kao što ste ranije naučili obrađivati takve podatke ima smisla jedino koristeći algoritme za kompresiju
- Teoretski kad razmatramo kompresija je super....no ako je želimo izvesti onda se susrećemo sa tipičnim inženjerskim problemima
- Pogledajmo još jednom kako izgledaju dijagrami obrade podataka za kompresiju slike i videa, no sada ćemo pokušati predvidjeti kompleksnost nekih blokova koje na takvim dijagramima postoje

JPEG analiza



MPEG koder


Reg.
Frame Mem.
+
DCT
Q
ENC
Buffer
Pre Proces.
Q-1
IDCT
+
MC
Frame Mem.
ME

Kako to implementirati u SW ili HW

- Najzahtjevnije operacije pri obradi slike/videa
  - Procjena/predviđanje pokreta !!!!!....!!!
  - DCT
  - RGB-YUV transformacija
  - Entropijsko kodiranje

Analiza izvedbe JPEG algoritma

- Iz tablica koje smo ranije proučili vidljivo je da je podataka kod slika i videa puno... no mi mislimo da su naši današnji procesori brzi...
- Pogledajmo načelno koliko su u stvari brzi...
  - (NAPOMENA: u okviru ovog predmeta nećemo se baviti analizom efikasnosti prevoditelja, OS-a i razvojne okoline. No zapamtite da efikasnost rješenja ENORMNO ovisi o ovim a i nekim dodatnim uvjetima.)
- Napravimo jednostavan program u C-u koji neće raditi ništa osim učitati podatke iz datoteke sa diska te svakom podatku npr dodati neku konstantu ...
- Primjer

Primjeri

- Što smo vidjeli na prethodnom primjeru...
  - Puno toga se dešava što ne znamo...ali izgleda dosta brzo...
- Možemo uočiti kolika je osnovna brzina izvođenja ovakvog jednostavnog programa
- Detaljnom analizom mogli bi ustanoviti precizno vrijeme izvođenja no to možemo i pojednostavniti na način da program izvedemo u petlji i dobijemo prosječno vrijeme

Konverzija prostora boja

- Kao što ste naučili RGB način zapisa podataka za sliku nije dobar za kompresiju podataka već se za to koristi YUV (YCrCb) prostor
- Prije pokretanja kompresije slika u RBG zapisu konvertira se u YUV
- Inverzna transformacija obavlja se nakon dekompresije a prije prikaza na zaslonu

Konverzija prostora boja

- RGB-YUV konverzija vrlo se lako može obaviti jednostavnom matričnom operacijom:

$$Y = (0{,}257*R) + (0{,}504*G) + (0{,}098*B) + 16$$
$$U = -(0{,}148*R) - (0{,}291*G) + (0{,}439*B) + 128$$
$$V = (0{,}439*R) - (0{,}368*G) - (0{,}071*B) + 128$$

- Vidimo da za izračun svakog piksela treba 9 množenja i 9 zbrajanja (+dohvat,spremanje)

Konverzija prostora boja

- Iako matematički trivijalna, konverzija boja jedan je od računalno zahtjevnijih zadataka pri kompresiji
- Ako izračunamo koliko je potrebno da se obradi jedna slika 1280x1024:
  - 11,9M množenja
  - 11,9M zbrajanja
  - +dohvat, spremanje
- Pogledajmo na primjeru koliko to traje:
  - Primjer
  - RGB2YUV v.1. AVG ≈ 405

Konverzija prostora boja

- Verzija 2: osnovna aproximacija

Y = (0.257 * R) + (0.504 * G);

U = - (0.291 * G) + (0.439 * B) + 128;

V = (0.439 * R) - (0.368 * G) + 128;

  - Primjer
  - RGB2YUV v.2. AVG ≈ 370

- Verzija 3: poboljšana aproximacija

data[i++] = (byte)(R/4 + G/2);

data[i++] = (byte)(-(G/4) + B/4 + 128);

data[i++] = (byte)(R/4 - (G /4) + 128);

  - Primjer
  - RGB2YUV v.3. AVG ≈ 43 **!!!!**

Analiza

- Za jednu vrlo zahtjevnu operaciju uspjeli smo značajno smanjiti procesorske zahtjeve
- No to smo postigli uz **degradaciju** kvalitete izračuna podataka
- Da li je kvaliteta zadovoljavajuća ili ne vrlo je teško empirijski definirati te je zato potrebno napraviti mnogo testova i subjektivnih provjera

Zadatak za vježbu (neobavezno)

- Korištenjem bilo kojeg alata (Visual Studio, Matlab, Mathematica,....) usporedite kvalitetu tri ranije opisane metode konverzije prostora boja
- Postupak:
  - Napišite funkciju koja učitava sliku u boji u RGB
  - Napišite tri različite funkcije transformacije boja (od kojih je jedna po punim formulama)
  - Izračunajte MSE za svaku komponentu (Y,U,V) za jednu testnu sliku za dva aproksimativna rješenja
  - Testnu sliku možete preuzeti na WEB-u

Načini optimizacije algoritama

- Vidjeli smo prvi primjer jedne jednostavne metode za optimizaciju nekog algoritma
- Iako se može činiti da je ovo dobar pristup njega na žalost ne možemo primjeniti u većini situacija
- Ponekad nije dozvoljeno unošenje ovako značajnih grešaka u izračune no ipak se računalni zahtjevi moraju značajno smanjiti

Načini optimizacije algoritama

- Neke osnovne grupe optimizacija:
  - Smanjenje preciznosti izračuna
  - Razvoj ekvivalentnih matematičkih algoritama sa manjom kompleksnosti
  - Promjena programske razvojne okoline
  - Promjena programske izvedbene okoline
  - Promjena arhitekture sustava za izvođenje

Načini optimizacije algoritama

- U projektiranju visokoefikasnih proizvoda morati ćemo se vrlo često poslužiti SVIM dostupnim metodama i njihovim kombinacijama
- U nastavku ćemo proučiti na koji način se može pristupiti optimizaciji i izvedbi nekih ključnih dijelova multimedijskih algoritama

Načini optimizacije algoritama

- Da bi mogli razmotriti moguće načine optimiranja MORAMO DOBRO POZNAVATI:
  - Algoritme koje optimiramo
  - Programska rješenja koja koristimo
  - Arhitekturu sustava na kojem se algoritmi izvode
- Mi ćemo upravo pokušati prema ovoj podjeli razmatrati multimedijske algoritme

Podjela osnovnih algoritama

Najvažniji algoritmi

Bez gubitaka

S gubicima

Statistički	**Univerzalni**	**U vremenskoj domeni**	**U frekvencijskoj domeni**
Shannon-Fano	**Duljina niza**	**Skalana kvanitzacija**	**Transformacijski**
Huffman	**Predviđanje**	**Vektorska kvantizacija**	**Filterski**
Aritmetičko kodiranje	**Metode rječnika**	**Kodiranje razlika**	**Wavelet**
Ostali	**Ostali**	**Ostali**	**Ostali**

Algoritmi bez gubitaka

- Dekompresirani podaci istovjetni originalu
- Statistički i univerzalni
- Manji omjeri kompresije
- Primjene: najvažniji podaci, medicina, financijska izvješća, ugovori, ...

Bez gubitaka – Statistički model

- Zasnovani na statističkim karakteristikama ulaznog niza simbola
  - Shannon-Fano
  - Huffman
  - Aritmetičko kodiranje
  - ...
- Više o algoritmima
  - compress.rasip.fer.hr
  - Puno web sjedišta

Shannon - Fano

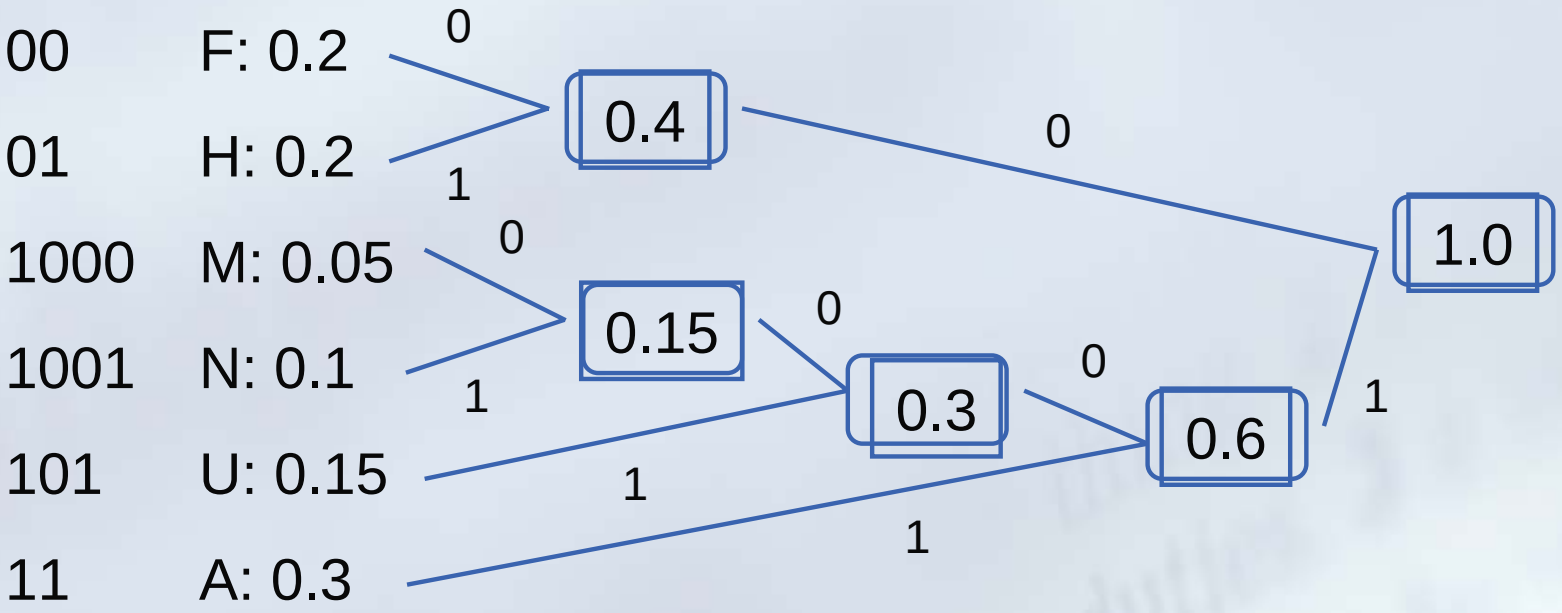
- Shannon-Fano algoritam

Huffman-ov algoritam

- Huffman D., metoda objavljena 1952.
- Zasnovan na statističkim svojstvima
- Dokazano najkraći kod promjenjive duljine (cjelobrojne duljine)
- Danas najčešće korišten kod za entropijsko kodiranje
- Ako se koristi nad podacima sa različitom distribucijom vjerojatnosti može povećati količinu podataka

Huffman-ov algoritam

Primjer kodiranja:



HUFFMAN = 01101000010001111001

56 bitova 19 bitova

Aritmetičko kodiranje

- Aritmetičko kodiranje-algoritam

Bez gubitaka - Univerzalni

- Sami se prilagođavaju statistici ulaznog niza
  - Duljina niza (Run Length)
  - Predviđanje
  - Metode rječnika

Run-Length algoritam

- Zasniva se na uzastopnom pojavljivanju ulaznih *simbola*
- Vrlo jednostavan za izvedbu
- Koristi se npr. za kompresiju slike u faksimil uređajima

Run-Length algoritam

- Primjer:
  - linije očitane sa dokumenta u fax uređaju

1 linija (75 dpi, 8”)=600 bita=75 B

000000000000000001111111111111111111111111000000000000.....000111111111.....11111000000

17,24,3,211,22,188,77,54,4 = 9 B

Predviđanje

- Primjer: **Stanje cijena dionica na burzi**

- Funkcija predviđanja
  - $x' = xt\text{-}1 + (xt\text{-}1 - xt\text{-}2)$

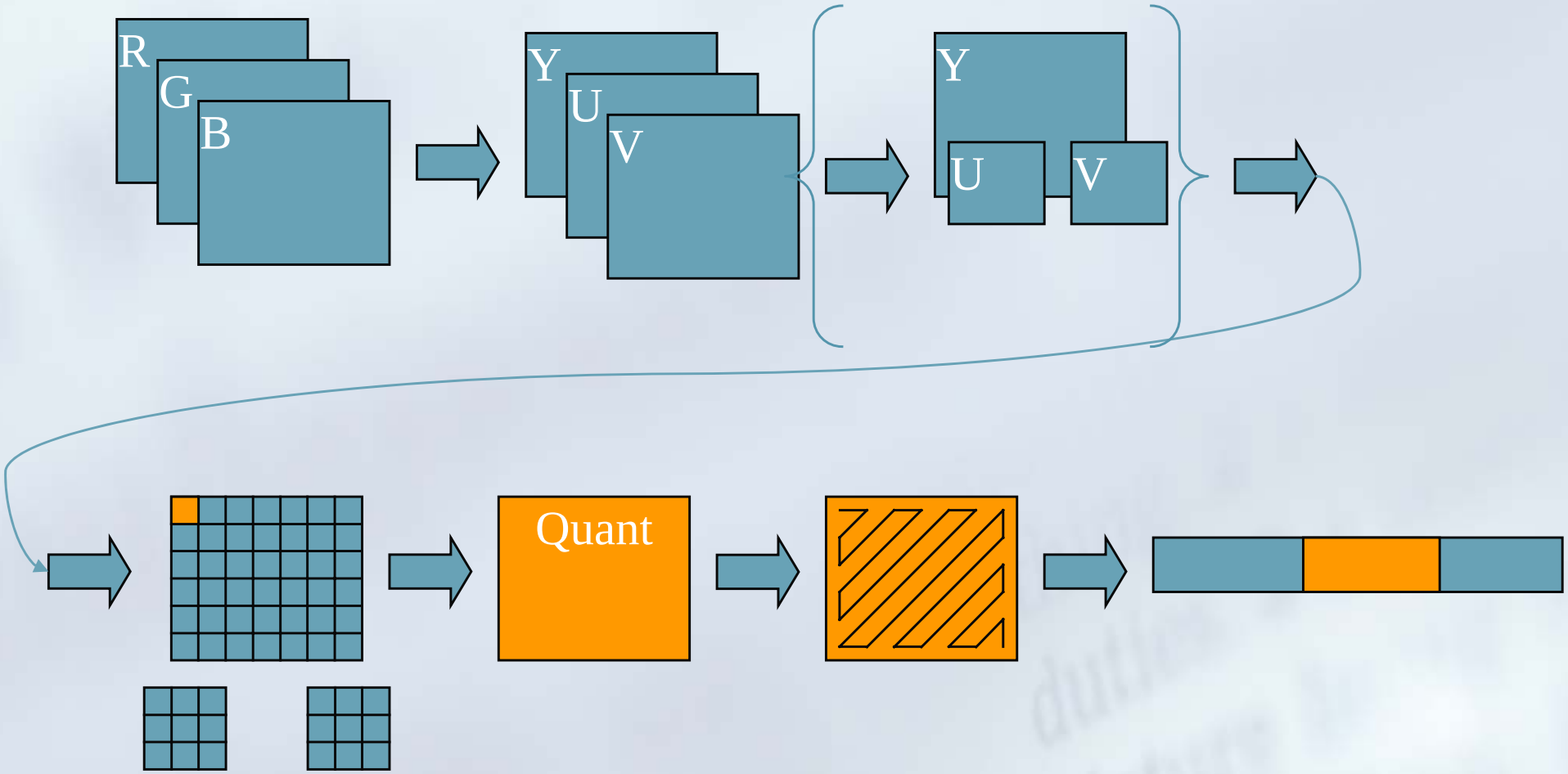
Predviđanje

Stock price	Predicted	Difference
134	0	134
141	0	141
145	148	-3
150	149	1
153	155	-2
152	156	-4
150	151	-1
...		

Predviđanje + Huffman

- Predviđanje:
  - dobija se nova, vrlo dobra raspodjela (često pojavljivanje malih vrijednosti, rijetko pojavljivanje velikih)
- Huffman
  - dobra kompresija za raspodjelu sličnu gore navedenoj
- Predviđanje+Huffman
  - Velika efikasnost kompresije

JPEG analiza



- Proučimo ponovo naš JPEG dijagram i pogledajmo gdje su najzahtjevniji odsječci

JPEG

- RGB2YUV:
  - Definitivno velika količina podataka
  - Osnovna metoda: 9*, 9+ po pikselu
  - (1280x1240) ≈ 10^7 množenja, 10^7 zbrajanja
- Vidjeli smo jednu metodu za smanjenje kompleksnosti ove konverzije

JPEG

- 2D-DCT, 2D-IDCT

$$C(u,v) = \alpha(u)\alpha(v) \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} f(x,y) cos(((2x+1)u\pi)/2N) cos(((2y+1)v\pi)/2N)$$

$$f(x,y) = \sum_{u=0}^{N-1} \sum_{v=0}^{N-1} \alpha(u)\alpha(v) C(u,v) cos(((2x+1)u\pi)/2N) cos(((2y+1)v\pi)/2N)$$

- ...pokušajmo proučiti koliko operacija po 2D-DCT elementu...

JPEG - DCT

$$C(u,v) = \alpha(u)\alpha(v) \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} f(x,y) cos(((2x+1)u\pi)/2N) cos(((2y+1)v\pi)/2N) \qquad \alpha(u) = \begin{cases} \sqrt{(1/N)} & \text{for } u = 0 \\ \sqrt{(2/N)} & \text{for } u = 1, 2, ..., N-1 \end{cases}$$

- Za svaki element:
  - α()α() ΣΣ f*cos()*cos()
  - Teoretski 66 množenja, 63 zbrajanja, 128 izračuna cos() funkcije (koja samo u parametrima ima 1 dijeljenje, 3 množenja, 1 zbrajanje)
  - Pokušajte ovo izračunati u bilo kojem programskom jeziku.....

JPEG - DCT

$$C(u,v) = \alpha(u)\alpha(v) \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} f(x,y) cos(((2x+1)u\pi)/2N) cos(((2y+1)v\pi)/2N) \qquad \alpha(u) = \begin{cases} \sqrt{(1/N)} & \text{for } u = 0 \\ \sqrt{(2/N)} & \text{for } u = 1, 2, ..., N-1 \end{cases}$$

- Primjer približne kompleksnosti računanja po teorijskoj formuli...
- Primjer
- Očito je da ovo ovako nije iskoristivo ni za kakve primjene....

JPEG - DCT

$$C(u,v) = \alpha(u)\alpha(v) \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} f(x,y) cos(((2x+1)u\pi)/2N) cos(((2y+1)v\pi)/2N) \qquad \alpha(u) = \begin{cases} \sqrt{(1/N)} & \text{for } u = 0 \\ \sqrt{(2/N)} & \text{for } u = 1, 2, ..., N-1 \end{cases}$$

- Postoji mnogo različitih matematičkih i računalnih metoda kako efikasno izračunati ovakvu funkciju
- Pogledati ćemo neke najvažnjije
- Odvojivost (separability), lookup tablice,...

JPEG – 2D-DCT odvojivost

$$C(u,v) = \alpha(u)\alpha(v) \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} f(x,y) cos(((2x+1)u\pi)/2N) cos(((2y+1)v\pi)/2N) \qquad \alpha(u) = \begin{cases} \sqrt{(1/N)} & \text{for } u = 0 \\ \sqrt{(2/N)} & \text{for } u = 1, 2, ..., N-1 \end{cases}$$

- Matematički možemo dokazati (ne u okviru ovog kolegija) da se 2D DCT može izračunati korištenjem 1D DCT koja se računa po retcima a onda po stupcima

- Formula 1D DCT

$$C(u) = \alpha(u) \sum_{x=0}^{N-1} f(x) cos(((2x+1)u\pi)/2N)$$

1D DCT

$$C(u) = \alpha(u) \sum_{x=0}^{N-1} f(x) cos(((2x+1)u\pi)/2N)$$

- Ako pogledamo kompleksnost vidimo da je približna kompleksnost ove formule za svaki element:
  - α() Σ f*cos()
  - Teoretski 9 množenja, 7 zbrajanja, 8 izračuna cos() funkcije (koja u parametrima ima 1 dijeljenje, 3 množenja, 1 zbrajanje)

2D DCT preko 1D DCT

$$C(u) = \alpha(u) \sum_{x=0}^{N-1} f(x) cos(((2x+1)u\pi)/2N)$$

- Priličan problem je cos() funkcija
- S obzirom na diskretizirane vrijednosti parametara cos() faktori se NE računaju već se koriste unaprijed izračunate vrijednosti pohranjene u tablicu (tzv.lookup tablica) te se kao takvi oni obično i kombiniraju sa α() parametrom.
- Samo ovim postupkom ZNAČAJNO smo smanjili kompleksnost računanja

2D DCT preko 1D DCT

$$C(u) = \alpha(u) \sum_{x=0}^{N-1} f(x) cos(((2x+1)u\pi)/2N)$$

- No i sa tako izvedenim cos() kompleksnost je velika:
  - 2D DCT za blok 8x8:
    - 4096 množenja, 4032 zbrajanja
- Samo za jednu sliku 1280x1024
  - 83886080 množenja, 82575360 zbrajanja

2D DCT preko 1D DCT

$$C(u) = \alpha(u) \sum_{x=0}^{N-1} f(x) cos(((2x+1)u\pi)/2N)$$

Koristeći svojstvo odvojivosti

- 1D DCT za 8 podataka
  - 64 množenja, 56 zbrajanja
- 2D DCT preko 1D DCT
  - 1024 množenja, 896 zbrajanja
- Samo za jednu sliku 1280x1024
  - 20971520 množenja, 18350080 zbrajanja

2D DCT

- No i ovo što smo do sada predložili nije dovoljno da u stvarnom svijetu izvedete neki algoritam za kompresiju na nekom manjem računalu
- Zato se računanju 2D DCT pristupa preko izračuna skalirane brze DFT (ponovo se koristi svojstvo odvojivosti)

Brzi algoritmi – što se koristi

- Tipično se za izračun 1D DCT preko 1D FFT koristi teorija:
  - AAN algoritam (1988) za skalirani 1D DCT (8 točaka):
    - 5 množenja, 29 zbrajanja, 16 dvojnih komplementa
  - Kovač, Ranganathan algoritam (1995) za skalirani 1D DCT (8 točaka):
    - 5 množenja, 29 zbrajanja, 12 dvojnih komplementa

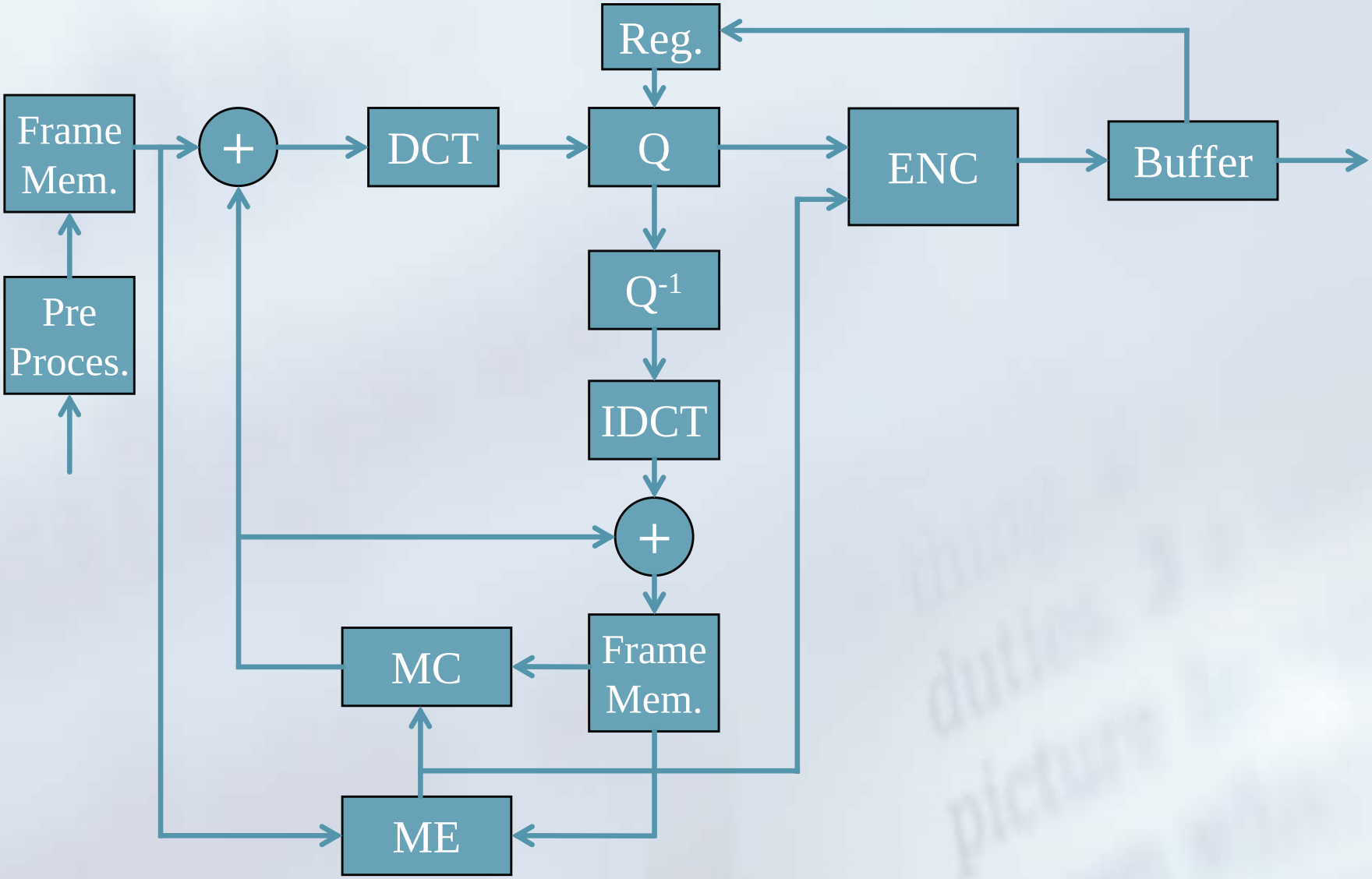
Kako se uistinu računa....

- Korištenjem ovih brzih algoritama za jednu sliku 1280x1024 potrebno je približno:
  - 1638400 množenja
  - 9502720 zbrajanja
  - 3932160 dvojnih komplementa
- Najvažnije je da je broj operacija množenja (SPORO u procesoru) smanjen cca. 15 puta

Zadatak za vježbu (neobavezno)

- Napišite program koji će računati 2D DCT za neku ulaznu sliku koristeći osnovni (teorijski) algoritam i koristeći odvojivost te 1D DCT

MPEG koder


Reg.
Frame Mem.
+
DCT
Q
ENC
Buffer
Pre Proces.
Q-1
IDCT
+
MC
Frame Mem.
ME