# 4. Komprese obrazu a videa, metody LZW, JPEG, MPEG.

**Komprese obrazu, úvod**

**Cíl**: Zmenšení množství dat potřebných k reprezentaci obrazu. Spotřebované množství paměti se měří například v bitech.

**Použití**: Pro přenos a uchování dat.

Proč se liší komprese 2D obrazu od komprese 1D dat?

Digitalizovaný obraz se chápe jako dvojrozměrná struktura (matice) náhodných vzorků.

Matematicky řečeno: Cílem je převést matici jasu (nebo 3 matice s barevnými složkami) do jiné reprezentace, která je méně statisticky závislá (zhruba řečeno korelovaná).

**Redundance a irelevance**

**Redundance v kódování**

Častěji se opakující symbol se kóduje kratším kódovým slovem. Optimální kódování: Huffmanovo a aritmetické kódování.

**Redundance mezi pixely**

Pomocí lineárních integrálních transformací obrazu, např. Fourierovou, kosinovou či vlnkovou transformací.

Prediktivní komprese.

Úsporné algoritmy generování obrazu, např. fraktální.

**Irelevance z hlediska vnímání člověkem**

Nezobrazit např. všechny jasové úrovně nebo vysoké frekvence (nedůležité zahodit).

**Rozdělení metod komprese obrazů**

**1. Segmentace objektu v obraze**, tj. s interpretací dat.

Metody jsou závislé na sémantice dat. Dosahuje se nejvyšších kompresních poměrů. Není možná zpětná rekonstrukce výchozího obrazu.

**2. Odstranění redundantní a irelevantní informace**, tj. bez interpretace dat.

Lze použít na libovolná obrazová data. Využívá se statistických nadbytečností v obraze (sekvenci obrazu).

**Redundance versus irelevance**

Dvě velké třídy používaných postupů:

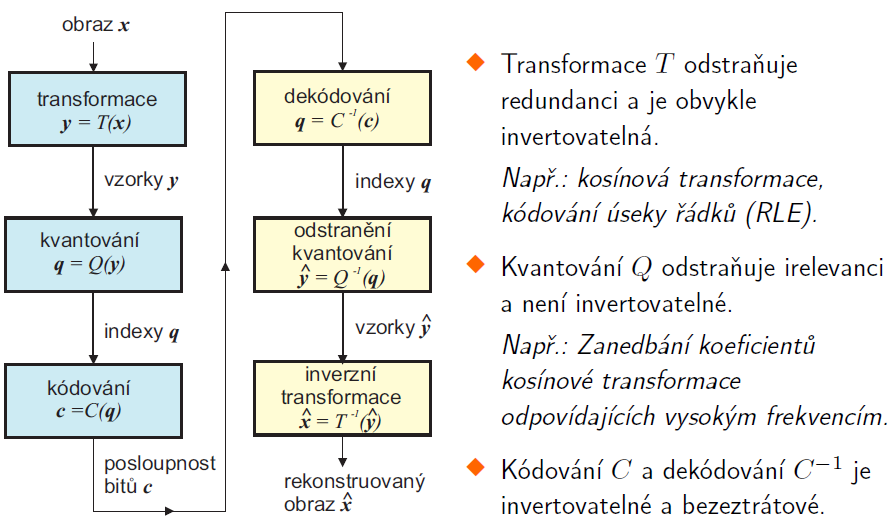
1. **Bezeztrátové metody**, odstraňují redundanci.

Odstraňují pouze statistickou nadbytečnost. Umožňují **úplnou** **rekonstrukci** výchozího signálu.

2. **Ztrátové metody**, odstraňují irelevantní informaci.

Nevratně odstraňují část informace, která má v daném kontextu malý význam (např. vysoké frekvence, jasové detaily nepostřehnutelné lidským okem). Umožňují pouze částečnou rekonstrukci výchozího signálu.

**Komprese obrazu a jeho rekonstrukce**



### Huffmanovo kódování

Huffmanovo kódování je algoritmus využívaný pro **bezeztrátovou** **kompresi dat**. Konvertuje znaky vstupního souboru do **bitových řetězců různé délky**. Znaky, které se ve vstupním souboru **vyskytují** **nejčastěji**, jsou konvertovány **do bitových řetězců s nejkratší délkou** (nejfrekventovanější znak tak může být konvertován do jediného bitu), znaky, které se vyskytují **velmi zřídka**, jsou konvertovány do **delších řetězců** (mohou být i delší než 8 bitů).

Nejjednodušší metoda komprese touto metodou probíhá ve dvou fázích. První projde soubor a vytvoří statistiku četností každého znaku. Ve druhé fázi se využije této statistiky pro vytvoření binárního stromu a k následné kompresi vstupních dat.

Dekomprese naopak pomocí rekonstruovaného binárního stromu dekóduje řetězce proměnlivé délky.

**Vstup:** symboly s pravděpodobností jejich výskytu; zpráva

**Výstup:** optimálně zakódovaná zpráva

**Postup:** podle pravděpodobností výskytu symbolu se zdola nahoru vytváří binární (Huffmanův) strom. Tento strom potom slouží ke generování zakódované zprávy. Prefixový kód, tj. žádné kódové slovo nemůže být prefixem žádného jiného kódového slova. Umožňuje dekódování, aniž by se znala délka jednotlivých slov.

* Pevný počet bitů na symbol.
* Nechť b je průměrný počet bitu na symbol. Potom průměrná délka kódového slova je L.
* Kódování je založeno na použití různě dlouhých bitových kódů pro symboly s různou frekvencí výskytu. Tudíž využívá podobný princip jako Morseova abeceda, kde nejvíce se vyskytujícím hláskám je přiřazen nejkratší kód.

Huffmanův kód má dvě důležité vlastnosti. Jednak je kódem s minimální délkou, jednak je to prefixový kód a je tedy jednoznačně dekódovatelný. Jeho problémem je to, že musíme znát rozdělení pravděpodobnosti výskytu jednotlivých symbolů. To lze nahradit odhadem, případně je možné tento odhad v průběhu komprese upřesňovat.

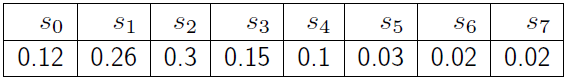
**Použití:** Huffmanova kódu je časté v kombinaci s jinými kompresními algoritmy, například při kompresi obrazu a videa ve standardech **JPEG a MPEG**.

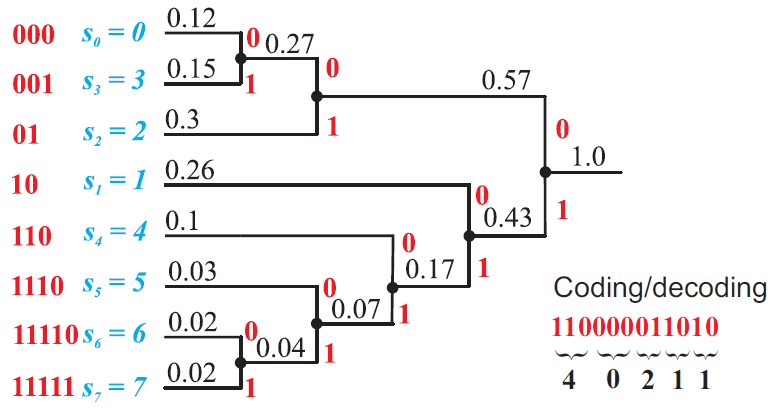
Samostatně se s ním můžeme setkat v programu compress pod OS Unix.

Přirozeně se teď nabízí otázka, zda je Shannon-Fanův kód optimální (Tento kód je tedy konstruován od kořene k listům a na rozdíl od Huffmanova kódování, jehož kód je tvořen od listů ke kořenu, nemusí být optimální.), zda někdy neexistuje prefixový strom, který by byl lepší. Odpověď na tuto otázku je ano. Shannon-Fanův kód nám skutečně nezajišťuje optimální řešení (tedy ani optimální kompresi) a jak vás už možná napadlo, tak algoritmus, který vždy vydá optimální prefixový strom je Huffmanův kód.

**Příklad**:

Vstup: zprávy obsahují osm symbolu s pravděpodobnostmi výskytu.

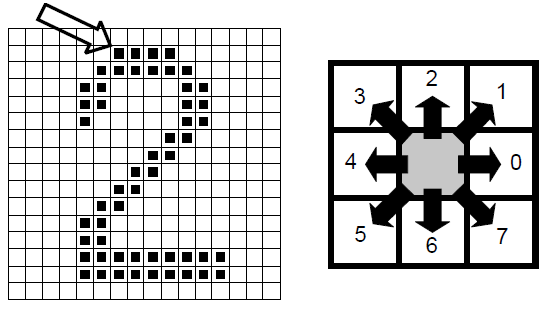




Vždy se k sobě sčítají nejnižší hodnoty, zde je chyba. (*obrázek z přednášky*)

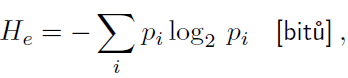
**Kódování segmentovaných dat**

**Kódování hranic oblastí,** Řetězový (též Freemanův) kód, 8-okolí



Kód: 00077665555556600000006444444442221111112234445652211

Entropie v teorii informace, Claude Shannon, 1948

kde *pi* je pravděpodobnost i-tého symbolu ve zprávě.

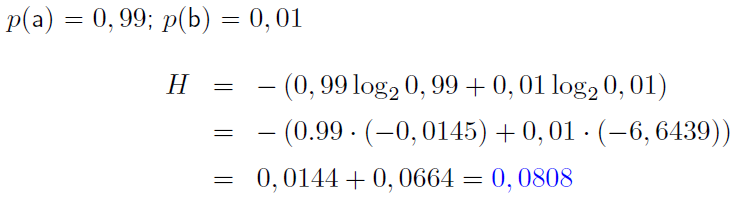
Je to množství informace v bitech, které poskytuje zdroj, zdrojem informací může být např. věta ze znaků, nebo házení kostkou apod. Vyjadřuje to, jakým nejmenším množstvím bitů jsi schopen tu větu třeba popsat.

**Příklad**:

Nechť jsou ve zprávě jen dva znaky a, b. 

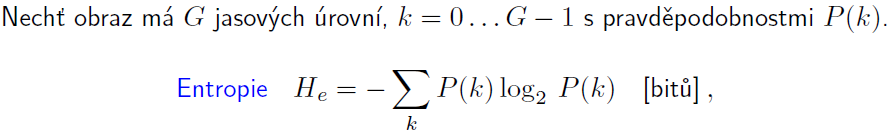


**Poznámka**: Takže tam, kde a a b mají stejnou pravděpodobnost výskytu je entropie **vysoká**, protože nemůžeš odhadnout, jaké písmenko bude následovat. Je to v bitech a bude 0, b bude 1. Takže 1 bit ti stačí na určení jestli bude následovat a nebo b. Vynásobíš počtem znaků ve větě a máš celou informaci věty.



**Poznámka**: Nízká entropie, protože a se vyskytuje často, hodně často. V podstatě kdybys řekl, že ve větě jsou samá áčka, tak bys chybu neudělal, udělal, ale malou. Prostě vědět, že čím jsou si pravděpodobnosti víc blízké, tím vyšší entropie.

**Entropie pro šedotónový obraz**



Nechť *b* je nejmenší počet bitů, kterým lze reprezentovat počet kvantizačních úrovní.

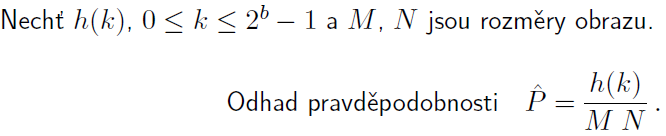


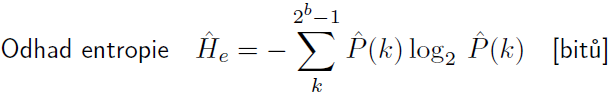
Na obrázku entropii odhaduješ z histogramu jasů. Udává kolik je v obrazu tam informace, ale je to pouze odhad. Takže když bude jednobarevný obrázek, tak to bude málo informace, když tam bude spousta barev (nebo jasových změn) tak bude entropie vysoká.

Tohle můžeš využít třeba u nějakého rozpoznávání v obrázcích.. budeš projíždět okénkem a když je tam málo informace tak nic neděláš třeba.

Takže to co je navíc je redundance, ovšem důležité je, že z histogramu jasu je to pouze odhad, entropie může být nižší, protože histogram ti nic neříká o tom, které pixely jsou u sebe a které ne. Je pravděpodobnější, že vedle světlého pixelu bude jiný světlý pixel, ale tohle z histogramu nezjistíš, proto ti entropie nevyjde taková jaká by měla vyjít skutečná, pouze odhad.

**Odhad entropie z histogramu obrazu**

****

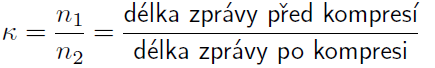
****

**Poznámka:**

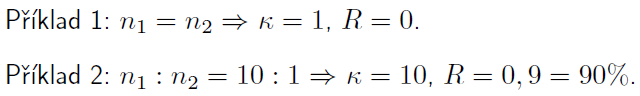
Odhad entropie je příliš optimistický. Ve skutečnosti je entropie nižší, protože mezi jasy jednotlivých pixelů obrazu existují statistické závislosti.

**Tři definice kompresního poměru**

1. Na základě redundance (měřené entropií) 

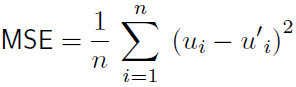
2. Na základě úspory paměti 

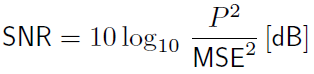
3. Relativní úspora paměti 



**Měření ztrátovosti komprese**

Necht je výchozí posloupnost a ztrátově komprimovaná posloupnost

**Středně kvadratická chyba** (Mean Square Error, MSE) 

**Poměr signálu k šumu** (Signal to noise ration, SNR) 

kde *P* je interval hodnot vstupní posloupnosti, 

**Špičkový poměr signálu k šumu** (peak-signal to noise ration, PSNR)



kde *M* je maximální interval hodnot vstupní posloupnosti, tj. 256 pro **osmibitový** **vstup** a 65356 pro **šestnáctibitový** **vstup**.

V praxi se používá zejména SNR a PSNR. Výraz MSE slouží jako pomocný pro jejich definici.

**Prediktivní komprese – myšlenka**

Najít matematický model, který dokáže predikovat na základě předchozích hodnot další hodnotu.

Přenášet pouze rozdíl mezi skutečnou a predikovanou hodnotou.

Ke kompresi dochází, protože rozdílová data mají menší statistickou variaci (např. rozptyl) než původní data.

Ztrátová komprese obrazu je založena na odstraňování irelevance dat. Je založena na pozorování vnímání člověka. Některé detaily je prakticky nemožné lidským okem vnímat, a proto je můžeme považovat za nadbytečné. Na rozdíl od bezeztrátové komprese nelze takto komprimovaný obraz přímo zrekonstruovat, část informace se nenávratně ztratí.

**[ JPEG**, JPEG 2000, **MPEG**, MP3, WMA, AAC, VQF ]

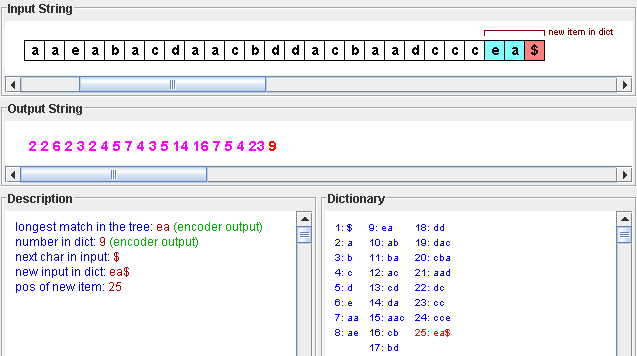
Bezeztrátová komprese obrazu odstraňuje redundanci dat. To znamená, že se odstraňuje nadbytečná informace, kterou je možné přímo rekonstruovat, tedy nedochází ke ztrátě dat.

[ GIF, PNG, TIFF, **JPEG** 2000 (zahrnuje bezeztrátovou kompresní metodu) ]

# LZW (Lempel, Ziv, Welch)

Kódovací algoritmus si postupně vytváří kódovací tabulku ze slov použitých v již zakódovaném textu. Tato tabulka mapuje vstup na slova/stringy s pevně stanovenou délkou. Na počátku je tabulka inicializována pomocí všech jednoznakových slov použité abecedy (typicky 256 znaků ASCII). A dále algoritmus sériově prohledává text, ukládá si do tabulky každé unikátní dvouznakové slovo jako zřetězení vzoru a kódu (něco jako slovník). Jakmile má uložena všechna dvouznaková slova, pošle na výstup kód prvního na vstupu. Algoritmus pokračuje v kódování, jakmile je na vstupu nalezeno již známé slovo (tj. je již v tabulce), na výstup se pošle odpovídající kódový znak plus před ním první znak kódovaného slova.

Dekompresnímu algoritmu stačí jen zkomprimovaný text, slovník si vytváří stejně jako u komprimace „za chodu“.



Čteš jedno písmenko a druhé (2 písmenka), takže aa, poslednímu písmenku říkejme třeba koncovka, takže máme a a za ním koncovku a, první a máme v tabulce, tak zapíšeme jeho index = 2. Dvojici aa nemáme v tabulce tak doplníme do tabulky a posuneme se. ae, nemáme v tabulce, tak zase doplníme a zapíšeme index ačka tedy 2. ea nemáme, zapíšeme do tabulky a doplníme index ečka. A takhle jedeme dál až narazíme na aa. aa máme v tabulce, tak vezmeš trojici aac a tu zapíšeš do tabulky, místo aa zapíšeš index 7 a pak zase pokračuješ po dvojicích a narazíš pak na da, které už je v tabulce, tak zase vezmeš trojici a trojici dáš do tabulky, místo da zapíšeš 14 a tak dále...

viz. <http://www.lodherov21.eu/projekty/data/others/logicke-programovani-program/pages/lzw-java-applet.html>

# JPEG

### Popis (Joint Photographic Expert Group. Standartizováno v roce 1992)

JPEG/JFIF je nejčastější formát používaný pro přenášení a ukládání fotografií na World Wide Webu. Není však vhodný pro perokresbu, zobrazení textu nebo ikonky, protože kompresní metoda JPEG vytváří v takovém obrazu viditelné a rušivé artefakty. Pro takové účely se většinou používají soubory PNG a GIF. Protože má GIF pouze 8 bitů na pixel, není vhodný pro barevné fotografie, PNG je možné použít pro ukládání fotografií, ale výsledná velikost souboru je nevhodná pro publikování na webu.

### Postup kódování

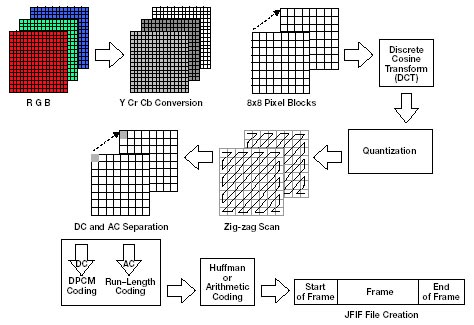
Barevné obrazy se nejdříve převedou z barevného prostoru RGB do prostoru YUV (model k popisu barvy používající tříprvkový vektor [Y,U,V], kde Y je jasová složka a U a V jsou barevné složky.), kde lze UV matice reprezentovat v polovičním rozlišení než matici Y (~~ intenzita).

**První generace** (.jpg) z 1992 používá DCT (diskrétní kosínovou transformaci) pro odstranění redundance a irelevance. Pro optimální kódování se použije převod koeficientu DCT do 1D vektoru, úseky řádku a symboly kóduje Huffmanovým kódováním.

**Druhá generace** JPEG2000 (.jp2) z roku 2000 odstraňuje redundanci a irelevanci pomocí vlnkové transformace. Potom kóduje v jednotlivých bitových rovinách a symboly kóduje aritmetickým kódováním.

Aby se ušetřil výpočetní čas, je obraz rozdělen na bloky 8 × 8, které jsou komprimovány nezávisle na sobě. Každý blok obrazu 8 × 8 lze vyjádřit jako lineární kombinaci bázových funkcí.

Výpočet DCT slouží k nalezení vah lineární kombinace, váhy jsou prahovány. Velikost prahu ovlivňuje míru komprese, tj. volí se irelevance.



**DCT, bázové funkce**

|  |  |
| --- | --- |
| Pevně zvolených 64 bázových funkcí.  Každý blok obrazu 8 *×* 8 lze vyjádřit jako lineární kombinaci bázových funkcí.  Výpočet DCT slouží k nalezení vah lineární kombinace.  Váhy jsou prahovány. Velikost prahu ovlivňuje míru komprese, tj. volí se  irelevance.  DCT2 base for [8x8] block |  |

**Poznámka:** DCT transformace, to jsou právě ty frekvence, když to procházíš zig zag pořadím, tak vybíráš nízké frekvence, dole vpravo jsou vysoké frekvence a to je stejně většinou **šum**, nebo drobné detaily, které se v té kompresi ztratí. DCT ti řekně kolik je kterých frekvencí v obrázku. Směrem doprava **roste** **frekvence** ve směru x a směrem dolů **roste** **frekvence** ve směru y.

Je to obdoba fourierovy transformace, výsledkem je matice 8x8 a představ si, že do každého toho čtverečku vepíšeš číslo, které bude říkat jak moc je tato frekvence důležitá. DCT ti pro každé okénko 8x8 vypočte matici čísel a ta čísla představují kolik které bázové funkce je v tom okénku přítomno. Ty frekvence jsou vždycky stejné.. co je různé jsou váhy se kterými pak budeš jednotlivé frekvence nasčítávat.

|  |  |
| --- | --- |
| Představ si, že ty obrázky z jednotlivých čtverců v matici budeš překládat přes sebe jeden přes druhý, dělat jejich součet, tak vezmu třeba první na pozici 1x1 a k němu přičtu druhý na pozici 2x1 a dostanu přechod od bílé do nějaké šedé. Představ si, že všech 64 obrázků přeložím přes sebe a nebudu je sčítat všechny stejně, ale budu přes nějaké váhy, řekněme první čtvereček s váhou 0,5, druhý čtvereček s váhou 0.01, třetí s váhou 0,8 a podle toho jaké váhy těm 64 obrázkům přiřadíš dostaneš různé obrázky výsledné. Řekl bych, že vybereš z toho jen čtverečky vlevo nahoře, protože ty nesou **nejvíce informací**.. a ty vpravo dole zahodíš... | Quantization matrix |

|  |  |  |
| --- | --- | --- |
| image block | image intensities | coefficients of the DCT2 |

# MPEG

|  |  |
| --- | --- |
| Název **MPEG** zkracuje anglická slova *Motion Picture Experts Group* **(Skupina expertů pro pohyblivý obraz)**, což je název pracovní skupiny vyvíjející standardy používané na kódování audiovizuálních informací (např. film, obraz, hudba) pomocí digitálního [kompresního](http://cs.wikipedia.org/wiki/Komprese_dat) algoritmu. |  |

Skupina MPEG standardizovala následující kompresní formáty:

* **MPEG-1**: Kódování pohyblivého obrazu a přidruženého zvuku pro digitální datové nosiče s rychlostí přenosu 0,9 až 1,5 Mbitu/s. Standard pro kódování zvuku zahrnuje také oblíbený zvukový kompresní formát Layer 3 (MP3).
* **MPEG-2**: Všeobecné kódování pohyblivého obrazu a přidruženého zvuku. Zahrnuje přenosové, obrazové a zvukové kódovací standardy pro vzduchem šířené televizní vysílaní ATSC a **DVB**, digitální satelitní TV přenos, digitální kabelový TV signál a (s určitými změnami) disky **DVD** **Video**. Přenosová rychlost se pohybuje od 1,5 Mbitu/s až do 15 Mbitů/s (pro TV signál se používá rychlost 6 Mbitů/s).
* **MPEG-3**: Původně určený pro kódování standardu HDTV, později byl jeho vývoj pozastaven a standard MPEG-3 byl sloučen se standardem MPEG-2.
* **MPEG-4**: Kódování audiovizuálního obsahu s velmi nízkým bitratem. Rozšiřuje formát MPEG-1 o podporu audio/video „objektů“, 3D obsahu, kódování s nízkou rychlostí přenosu a Digitální správu práv (angl. Digital Rights Management (DRM)).
* **MPEG-7**: standard pro popis dat s multimediálním obsahem, čímž se zcela odlišuje od předchozích (neříká totiž, jak data kódovat). Tento formát by měl sloužit k rychlému a efektivnímu vyhledávání multimediálních dat dle klíče. (ISO/IEC 15938)

**Princip:**

Kompresní formáty MPEG využívají převážně tzv. **ztrátovou** **kompresi** pomocí transformačních kodeků. U ztrátových transformačních kodeků se vzorky obrazu nebo zvuku rozdělí na drobné segmenty, transformují se na frekvenční prostor a poté kvantizují (quantized). Výsledná kvantizovaná data se dále kódují entropicky.

V rámci standardu MPEG je popsán jen formát bitového proudu a dekodér (angl. decoder). Kodér (angl. encoder) není v rámci standardu popsán. Pro členy skupiny MPEG jsou však k dispozici referenční implementace, které vytvářejí platné bitové proudy. To v praxi znamená, že libovolný dekodér formátu MPEG-4 dokáže dekódovat libovolný materiál formátu MPEG-4 (stejného typu) bez ohledu na to, jakým kodérem byl konkrétní materiál kódovaný.