Aplikace matematiky: komprese obrazu

Digitální fotografie jsou podstatnou součástí dat, která skladujeme, a proto je přirozené zkoumat, jak fotografie a obraz obecně efektivně skladovat s co nejmenší náročností na paměť zároveň se zachováním uspokojivé kvality.

Obraz, tedy dvourozměrná data lze přirozeně interpretovat pomocí matice **A**: každý bod-pixel digitalizované fotografie můžeme kódovat nějakým číslem, které určuje, jakou má daný pixel barvu. V případě černobílých obrázků si vystačíme s hodnotami 0-255 každý prvek naší matice tvoří jeden byte udávající stupeň šedi: 255 kóduje bílou, 0 černou. Pro barevné fotografie je nejjednodušší zvolit variantu kde místo matice **A** n x m volíme matici **A** n x m x 3 kde prvek a_{ij1} kóduje červenou prvek a_{ij2} zelenou a a_{ij3} modrou barvu (opět v rozsahu 0-255), výsledná barva je potom barva standartu RGB. Tedy na jeden pixel potřebujeme 3 byty paměti.

Jde nám tedy o nějakou dobrou reprezentaci matice **A** takovou, aby náročnost na data byla co nejmenší. Dále budeme chtít, aby bylo možné některá „nejméně důležitá" data matice **A** oříznout a ušetřit tak paměť ale neztratit mnoho informace, uložené v **A**. Pokud $\mathcal{A}$je oříznutí matice **A**, potom $|A - \mathcal{A}|$, kde $|\mathbf{X}|$ je norma určená největším singulárním číslem matice **X**, nám udává „míru komprese": o kolik informací jsme ochotni přijít výměnou za nižší nároky na paměť.

Všechny tyto naše požadavky splňuje SVD (singular value decomposition) neboli singulární rozklad matice. SVD je jedním z nejsilnějších a nejdůležitějších nástrojů maticových výpočtů, umožňuje určit hodnost či normu matice, ortogonální báze oboru hodnot a nulového prostoru matice apod.

<u>Definice:</u> Nechť **A** je matice typu n x m, odmocniny nenulových vlastních čísel matice $\mathbf{A}^T\mathbf{A}$ nazveme singulárními čísly matice **A**, $\sigma_j = \sqrt{\lambda_j}, j = 1, \dots, r;\ r = rank(A)$, je možné (a výhodné) uvažovat λ_j seřazený od největšího k nejmenšímu a tedy i σ_j jsou uspořádána odpovídajícím způsobem.

Označíme-li $u_j = \frac{\boldsymbol{A}v_j}{\sqrt{\lambda_j}}, v_j\ vektory\ ortonormální\ báze\ \ složené\ z\ vlastních\ vektorů\ \boldsymbol{A^TA}$ $příslušných\ nenulovým\ \lambda_j$ potom můžeme psát $\boldsymbol{A}v_j = \sqrt{\lambda_j} * u_j$, tedy $\boldsymbol{A}v_j = \sigma_j * u_j$ maticově lze tuto skutečnost zapsat jako $\boldsymbol{AV} = \boldsymbol{U\Sigma}$ kde **V**, **U** jsou matice tvořené vektory u_j a v_j a $\boldsymbol{\Sigma}$ je blokově diagonální matice typu n x m s blokem $\boldsymbol{\Sigma_r} = diag(\sigma_1, \dots, \sigma_r)$ na pozici 1,1, ostatní bloky jsou nulové. Tedy celkem $\boldsymbol{A} = \boldsymbol{U\Sigma V}^T$, navíc ukážeme, že takto zvolené matice **U** a **V** jsou ortogonální:

Matice $\mathbf{A}^T\mathbf{A}$ je symetrická pozitivně definitní tedy existují vlastní vektory v_j příslušné vlastním číslům λ_j které jsou navzájem ortonormální, tím jsme dokázali, že matice **V** lze zvolit ortonormální. Pro vektory u_j (odpovídající nenulovým vlastním číslům λ_j) platí:

$$< u_i; u_j > = < \frac{\boldsymbol{A}v_i}{\sqrt{\lambda_i}}; \frac{\boldsymbol{A}v_j}{\sqrt{\lambda_j}} > = \left(v_i^T \boldsymbol{A}^T\right)\boldsymbol{A}v_j \frac{1}{\sqrt{\lambda_i}\sqrt{\lambda_j}} = v_i^T(\boldsymbol{A}^T\boldsymbol{A}v_j)\frac{1}{\sqrt{\lambda_i}\sqrt{\lambda_j}} = v_i^T\lambda_j v_j \frac{1}{\sqrt{\lambda_i}\sqrt{\lambda_j}} =$$

$$= < v_i; v_j > \frac{\sqrt{\lambda_j}}{\sqrt{\lambda_i}} = 0\ pro\ i \neq j\ a = 1\ pro\ i = j$$

Tedy u_j tvoří ortonormální posloupnost a tu lze libovolně doplnit na ON bázi (za nulová vlastní čísla) a dostaneme ON matici **U.**

Věta: Pro každou matici **A** typu n x m hodnosti r existují unitární (ortogonální) matice **U** typu n x n a **V** typu m x m a kladná čísla $\sigma_1 \geq \cdots \geq \sigma_r > 0$ tak že $\boldsymbol{A} = \boldsymbol{U\Sigma V}^T$, kde vše je jako výše.

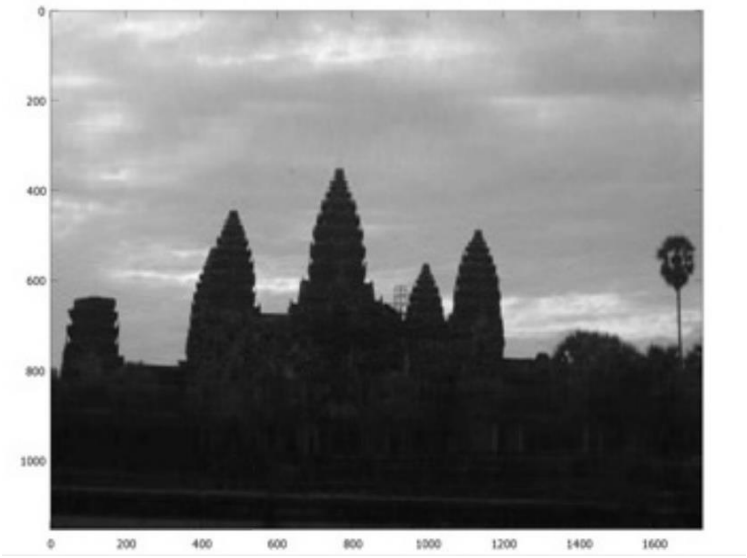
Rozklad $\boldsymbol{A} = \boldsymbol{U\Sigma V}^T$ nazveme singulárním rozkladem matice **A**.

Tento zápis je však pro praktické účely nevhodný, kvůli velikosti násobených matic a faktu, že $\boldsymbol{\Sigma}$ je téměř celá nulová, ekonomickým singulárním rozkladem nazveme rozklad $\boldsymbol{A} = \boldsymbol{U_r \Sigma_r V_r^T}$ kde **Ur**, **Vr** jsou matice typu n x r resp. m x r tvořené prvními r sloupci matic **U** a **V**. dále můžeme tento součin zapsat (díky tomu že $\boldsymbol{\Sigma_r}$ je diagonální) pomocí dyadického rozvoje: $\boldsymbol{A} = \sum_{j=1}^{r} \sigma_j u_j v_j^T$.

Použijeme singulární rozklad na naší matici **A** reprezentující fotografii: použití dyadického rozvoje potřebuje r*(1+n+m) dat pro černobílou fotografii, pro barevnou fotografii jednoduše uvažujeme 3 matice $\mathbf{A_R}$, $\mathbf{A_G}$, $\mathbf{A_B}$, pro každou ze 3 základních barev jednu, s příslušnými hodnostmi r_R, r_G, r_B potom každá z nich potřebuje r_R*(1+n+m) resp. r_G*(1+n+m), r_B*(1+n+m) dat, tedy celkem (r_R+r_G+r_B)*(1+n+m) dat.

Nyní si vzpomeneme, že singulární čísla byla seřazena podle velikosti. Změníme-li sumu $\boldsymbol{A} = \sum_{j=1}^{r} \sigma_j u_j v_j^T$ takž, že „zapomeneme malá singulární čísla“ dostaneme $\boldsymbol{\mathcal{A}} = \sum_{j=1}^{k} \sigma_j u_j v_j^T$ k<r která více méně zachovává informaci uloženou v **A**. V závislosti na k potom dostáváme různě přesné (a paměťově náročné) aproximace **A**. náš obrázek potom zabere pouze k*(1+n+m) dat pro černobílou variantu resp. (k_R+k_G+k_B)*(1+n+m) dat pro barevný obrázek. Takto zkomprimovaný obrázek zabere (k_R+k_G+k_B)/(r_R+r_G+r_B)*100% dat, úspora je tedy [1-(k_R+k_G+k_B)/(r_R+r_G+r_B)]*100% dat. Největším problémem ovšem je, jak určit k_R, k_G, k_B aby byl výsledek uspokojivý jak kvalitou, tak množstvím ušetřeného místa. Intuitivní je řídit se poměrem $p_i = \sigma_i : \sigma_1$, potom stačí zvolit p z intervalu (0,1) a k volit tak, že platí p_i < p pro všechna i větší než k. Takováto volba k je obzvláště výhodná, jsou-li singulární čísla rozložena nerovnoměrně (tedy **A** má několik málo výrazně větších singulárních čísel a ostatní jsou (velmi) malá, potom vezmeme pouze prvních pár členů v sumě a úspora dat je větší než kdybychom například volili k=r*q, q z (0,1) nebo má **A** všechna singulární čísla skoro stejně velká, potom nedojde k výrazné úspoře dat ale ani k nadměrné ztrátě kvality). Samozřejmě znalost počáteční hodnosti matice **A** případně rozložení singulárních čísel nám může výrazně pomoci s výběrem vhodného k. Zkomprimovaný obrázek pak dostaneme jednoduše vynásobením vektorů uvnitř sumy a sečtením, v podstatě se jedná o rozložení obrazu na jednotlivé „vrstvy“ a následné ořezání takových vrstev, které nesou nejmenší informaci o celkovém obrazu, přicházíme tedy o detaily, nicméně hlavní rysy zůstávají zachovány i pro velmi malá k, jak je vidět v následujících ukázkách.

Originální obrázek je velikosti 1728x1154 pixelů, tedy očekáváme r ≈ 1154. Pro k=13 dostaneme tuto aproximaci, která není velmi kvalitní, nicméně zabírá pouze 1.88% dat potřebných k uložení původní fotografie a jsou z ní rozpoznatelné hrubé rysy fotografie.

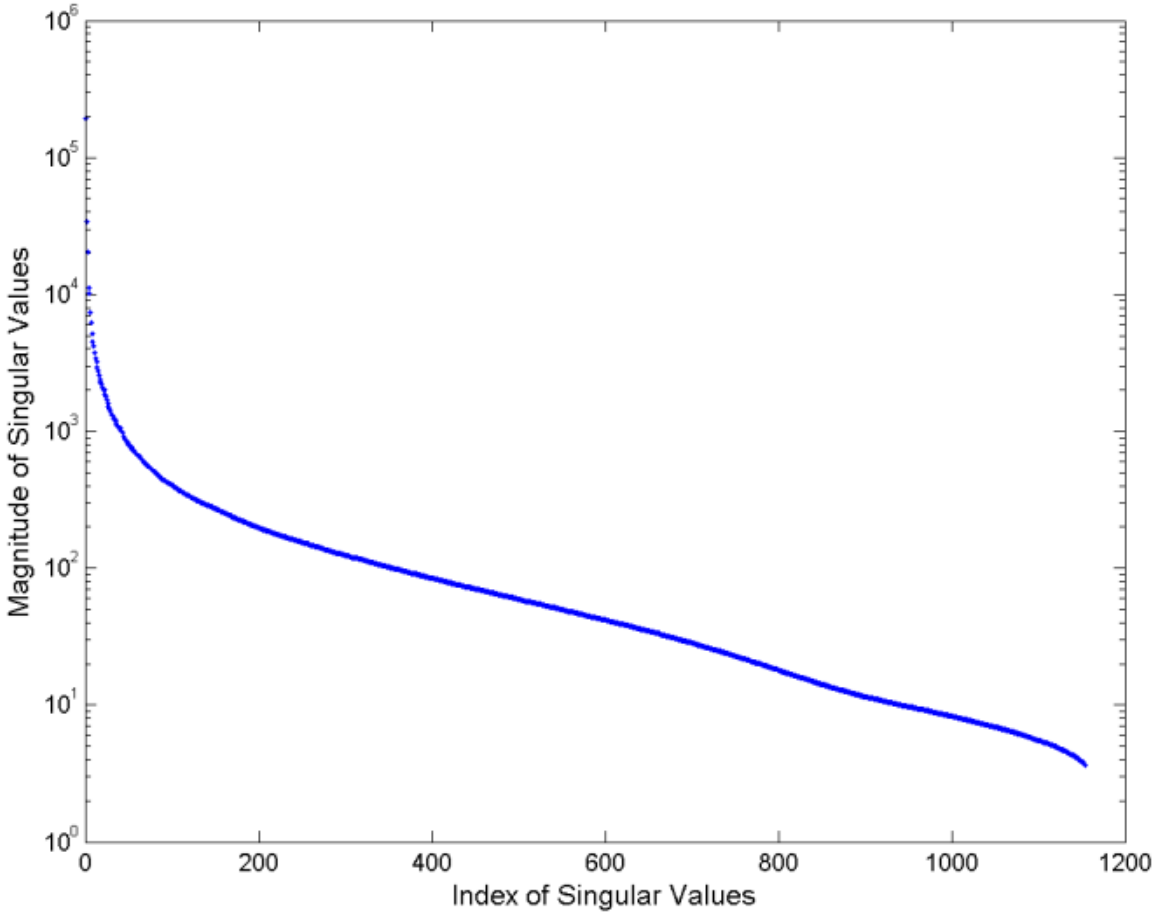


k=49, objevují se detaily

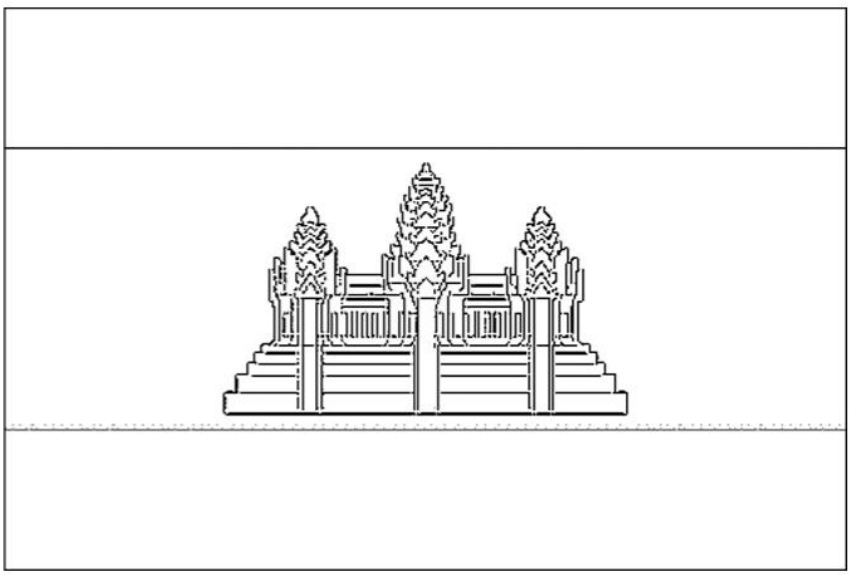
K=60, aproximace je v použitelném stavu a přitom zabírá pouze 8,68% paměti oproti originálnímu obrázku.

Originální obrázek, můžeme si všimnout, že došlo k celkovému ztmavení fotografie, to je způsobeno tím, že obrázek neobsahuje (ve spodní polovině) mnoho světlých bodů a tedy při

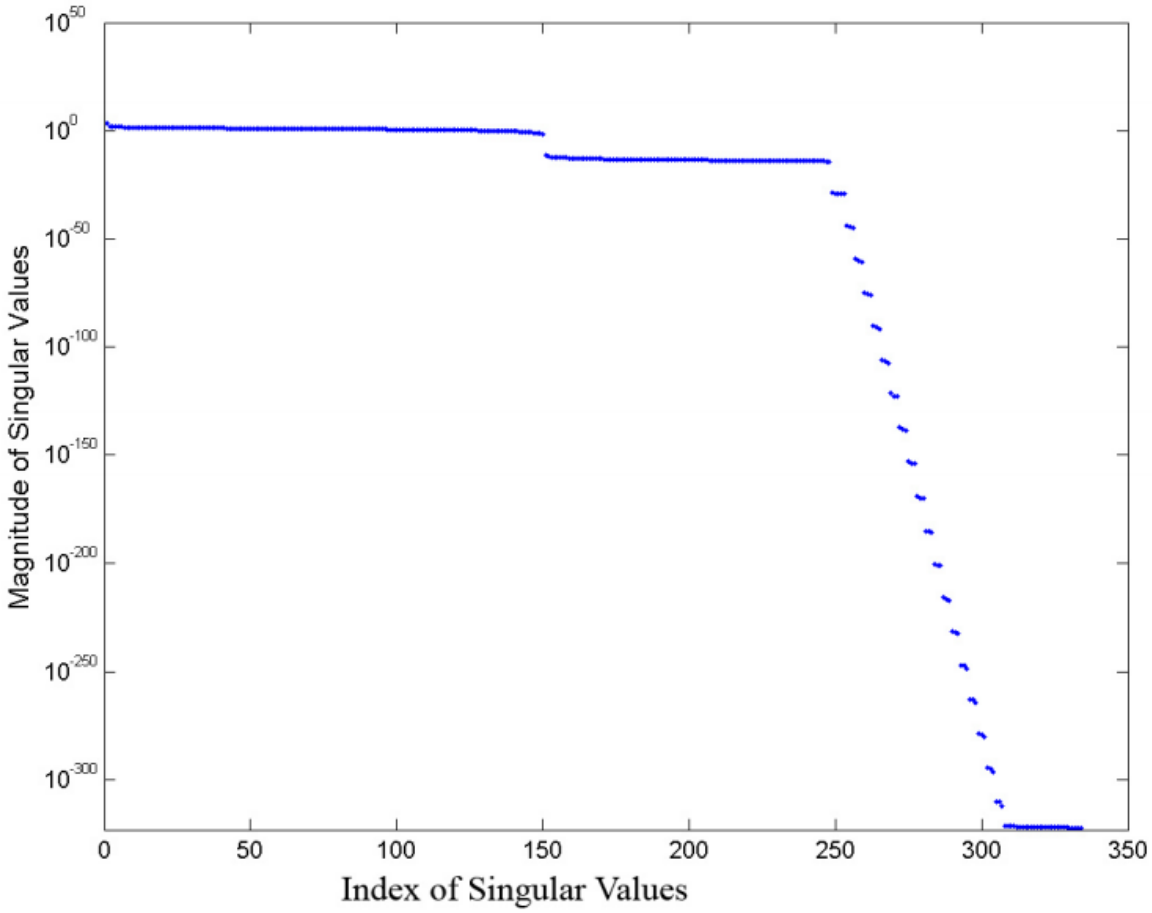
ořezávání se prosadily dominantnější, tmavší části. Celkově je tato fotografie dobře komprimovatelná, díky tomu, že **A** má několik málo velkých singulárních čísel a zbytek je zanedbatelný. Podívejme se na graf velikosti singulárních čísel:



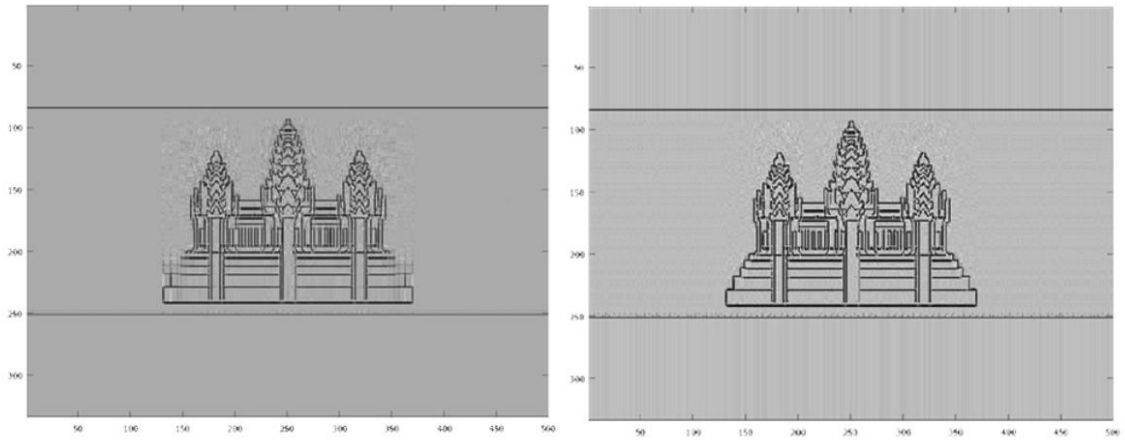
jde vidět, že řád prvních málo singulárních čísel je mezi 10^6 a 10^3, zbytek potom rychle klesá, od indexu 200 nemá prakticky cenu singulární čísla uvažovat (i když jsou hodnoty mezi 5 a 100). To přímo vysvětluje volbu hodnot k pro ukázkové komprese, použitelnost k =60 a absenci ukázek pro větší k. Podobný graf má většina „složitých“ objektů. Podíváme se na objekty jednoduché, k tomu nám poslouží čistě černobílý obrázek:



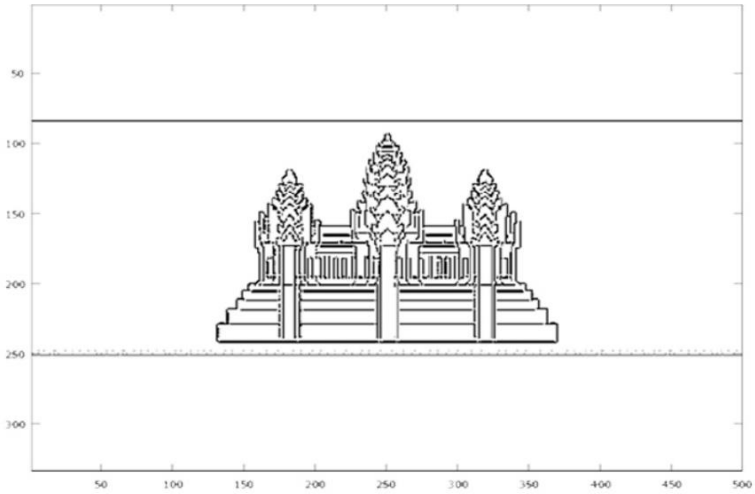
Všimneme si, že téměř celá informace je soustředěna uprostřed obrazu, zatímco okraje jsou prakticky prázdné, v řeči matic to znamená velké množství lineárně závislých sloupců a tedy malou hodnost. Podívejme se na graf singulárních čísel (obrázek má rozměry 500x334 pixelů):



Na první pohled vidíme, že prvních 150 singulárních čísel je prakticky stejně velkých, tedy za k bychom měli volit minimálně 150, zároveň jsou tato čísla poměrně malá (mezi 5 a 0,1), dalších 100 čísel (tedy 100 členů sumy) nám ještě nese nějakou informaci, nicméně řádově jsou tato čísla 0,0001 a menší, zbytek je již téměř nulový (pro strojovou aritmetiku jsou tato čísla skutečně rovna nule).



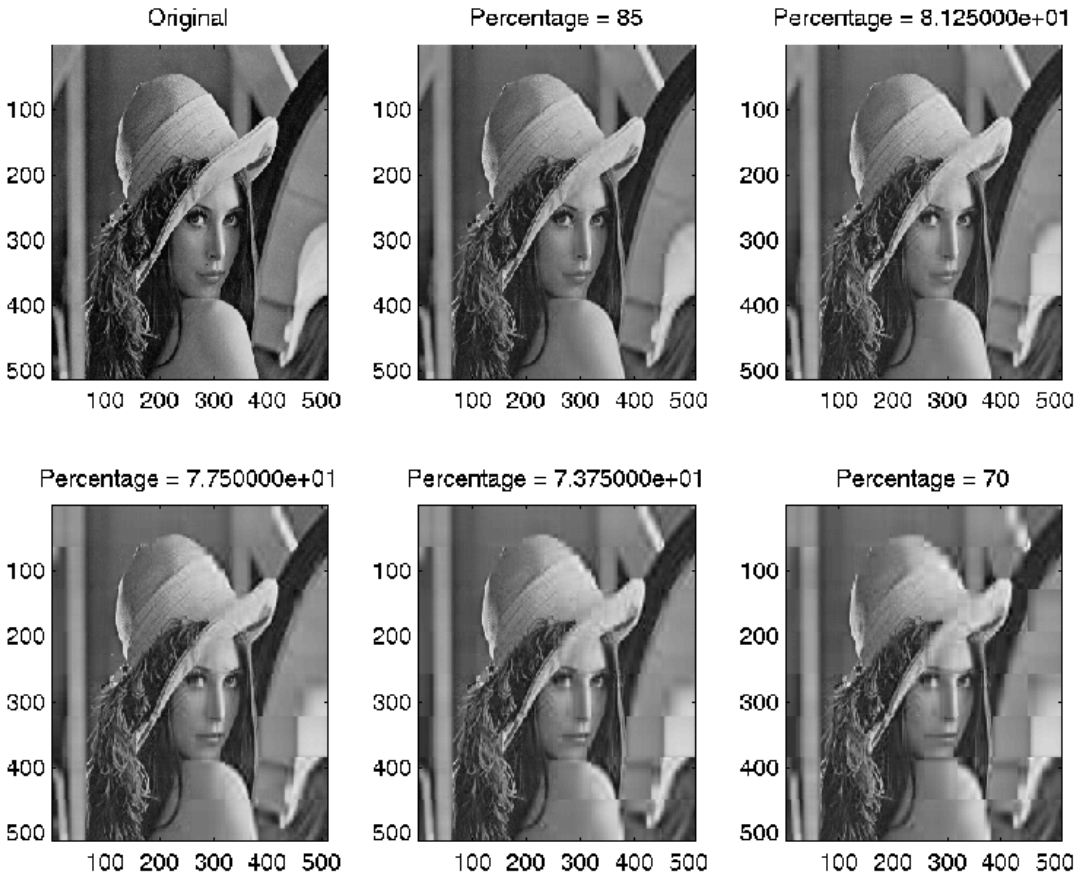
Aproximace pro k=30 a k=60, aproximace je viditelně špatná, nicméně jde poznat, co na obrázku je, úspora dat je 85% resp. 70%.



pro k=150 dostaneme podle očekávání kvalitní aproximaci nicméně úspora dat je pouze 25%.

Tím jsme demonstrovali výhodnost volby k podle předem stanoveného poměru p.

Existují různé modifikace SVD, které dále prohlubují úsporu dat při stejné kvalitě. Jednou z nich je například rozdělit původní obrázek na několik menších, a provést SVD na každém kousku zvlášť a nakonec je opět slepit dohromady. To je obzvláště výhodné, existují-li v obrázku plochy s málo detaily. Jednobarevnou plochu lze popsat pouze jedním singulárním číslem, tedy úspora je maximální možná, a nedochází ke ztrátě informace. Segmenty obsahující mnoho detailů budou komprimovány stejně (nebo dokonce lépe-přesněji) jako kdybychom obrázek nerozřezali. Nicméně tato metoda má své nevýhody – často jsou vidět přechody mezi jednotlivými segmenty (jak je demonstrováno na obrázku) :



Originál byl rozřezán na 64 částí, každá pak komprimována zvlášť percentage vyjadřuje, kolik z původních singulárních čísel bylo použito pro každou jednu z 64 částí. I když je aproximace velmi dobrá a zachovává detaily, už pro 77, 5% jdou vidět hrany jednotlivých bloků, což kazí celkový dojem z fotografie.

Výše popsaný způsob komprese má v současnosti spíše teoretické využití, v komerčních programech se využívá spíše algoritmů založených na rychlé Fourierově transformaci (FFT-fast Fourier transformation) který je pouze zrychlením diskrétní Fourierovy transformace z času $O(N^2)$ na O(N logN). Ta převádí zadanou posloupnost čísel na posloupnost frekvencí (složení sin a cos) předpisem:

$x_0, \dots, x_{N-1}$ posloupnost N komplexních čísel je transformována na N periodickou posloupnost

$$X_k = \sum_{n=0}^{N-1} x_n * e^{\frac{-i2\pi kn}{N}}, k \in \mathbb{Z}$$

Jelikož pracujeme pouze s reálnými čísly (representace obrazu pomocí matice zůstává jako v případě SVD) je výhodné použít diskrétní cosinovou transformaci (DCT), nutno ještě dodat, že x_k mohou být rovněž vektory, vícerozměrná (dvoudimenzionální) transformace je potom zopakování cosinové transformace jednou pro řádky a jednou pro sloupce:

$$X_k = \sum_{n=0}^{N-1} x_n * \cos\left[\frac{\pi}{N}\left(n + \frac{1}{2}\right)k\right], k = 0, \dots, N-1$$

Jedná se o tzv. DTC-II, která je nejhojněji používanou cosinovou transformací. Její odpovídající inverzní transformace je pak DCT-III přenásobená vhodnou konstantou:

$$X_k = \frac{1}{2}x_0 + \sum_{n=1}^{N-1} x_n * \cos\left[\frac{\pi}{N}n\left(k + \frac{1}{2}\right)\right], k = 0, \dots, N-1$$

Tato transformace přenásobená 2/N je potom inverzní transformací k DCT-II.

Popíšeme si, jak probíhá komprese používaná metodou JPEG, tu rozdělíme do několika kroků:

1. **Příprava obrazu:** obraz vyjádříme v maticovém tvaru, jak jsme si ukázali dříve, popřípadě použijeme rozklad barevných obrazů $Y'C_BC_R$ kde Y′ značí jas a C_B, C_R udávají barvu (rozdíl modré a červené složky) tento způsob vyjádření není nutný, pouze dále prohlubuje úsporu dat, Y totiž více méně representuje černobílou variantu obrazu a je podstatnější než zbylé dvě složky, které snesou silnější poměr komprese.
2. **Rozložení do bloků:** matice reprezentující obrázek rozložíme na bloky 8x8 pixelů, ty zpracováváme dále zvlášť.
3. **Cosinová transformace:** Každý blok 8x8 převedeme pomocí cosinové transformace na matici frekvencí. Nejprve si matici upravíme ta, aby hodnoty byly centrované kolem nuly, jelikož uvažujeme hodnoty jednotlivých pixelů v intervalu (0,255) odečteme od všech prvků matice hodnotu 128. Tento krok pomáhá snížit hodnoty, které se budou objevovat po cosinové transformaci. Nyní provedeme dvoudimenzionální cosinovou transformaci:

$$G_{u,v} = \frac{1}{4}\alpha(u,v)\sum_{x=0}^{7}\sum_{y=0}^{7} g_{x,y} * \cos\left[\frac{\pi}{8}\left(x + \frac{1}{2}\right)u\right]\cos\left[\frac{\pi}{8}\left(y + \frac{1}{2}\right)v\right]$$

Kde $G_{u,v}$ je hodnota na pozici u,v transformované matice (u,v jsou celá čísla od 0 do 7), $\alpha(u,v)$ je ½ pro u=v=0 a 1 jinak (spolu s ¼ tvoří koeficient, který zaručuje ortogonalitu vzniklé matice, $g_{x,y}$ je hodnota na pozici x,y v původní matici. Cosinová transformace má tendenci „hromadit informace do jednoho rohu matice“ tedy hodnota je nejvyšší (v absolutní hodnotě) na pozici 0,0 a má tendenci klesat s rostoucím součtem u+v. Hodnoty se zaokrouhlují na 2 desetinná místa. (hodnoty mohou být ve větším rozsahu, než 255 tedy je dočasně nutné zvětšit prostor, který matice zabírá přechodem k 16 bitům /pixel z 8bit/pixel

4. **Quantization:** lidské oko dobře rozpoznává změny na velkých plochách, ovšem změny světlosti na malé vzdálenosti nezachytí, informace o těchto malých změnách se transformací přenáší na prvky $G_{u,v}$, kde u,v jsou „velká“, tedy na menší čísla blíže pozice blíže pravému dolnímu rohu matice. To nám umožňuje kompresi dat. Každý prvek transformované matice podělíme nějakou předem danou konstantou (tyto matice **Q** 8x8 jsou uloženy v programu) a výsledek zaokrouhlíme k nejbližšímu celému číslu. Tím dostatečně zmenšíme prvky matice, aby šly opět popsat 8 bity a zároveň se nám větší počet prvků zaokrouhlí na nulu a tím dojde k úspoře dat. Poměr komprese je dán maticí Q, čím větší má prvky, tím více prvků $G_{u,v}$ bude zaokrouhleno na nulu a tedy se ušetří více místa. Zbylé nenulové prvky se navíc neukládají v matici ale v různých posloupnostech například postupně prvky na pozicích (0,0), (1,0),(0,1)(2,0),(1,1),(0,2) atd. dokud nenásledují samé nulové prvky.

Zpětná rekonstrukce: probíhá jednoduše, nejdřív se z posloupnosti sestaví matice 8x8, ta se přenásobí příslušnou maticí **Q**, na výslednou matici **F** se použije inverzní cosinová transformace:

$$f_{x,y} = \frac{1}{4}\sum_{u=0}^{7}\sum_{v=0}^{7}\alpha(u,v)F_{u,v} * \cos\left[\frac{\pi}{8}\left(x+\frac{1}{2}\right)u\right]\cos\left[\frac{\pi}{8}\left(y+\frac{1}{2}\right)v\right]$$

Potom už stačí jen ke každému prvku $f_{x,y}$ přičíst 128 a dostaneme rekonstruovaný obrázek.

Poměr komprese 1/15 (velikost rekonstruovaného obrazu ku původní velikosti) je velmi kvalitní:

<-Originál

Komprimovaný ->

Při poměru 1:46 se začíná objevovat typické „rozkostičkování“, projevuje se rozložení do 8x8 bloků, nicméně je pořád velmi dobře rozpoznat, co na obrázku je.

Právě typické rozkostičkování (artefakty) jsou příčinou potřeby jiné kompresní metody v těch případech, kdy záleží na kvalitě komprimovaných obrazů. Důležitým příkladem je zpracování otisků prstů. V devadesátých letech v Americe měla FBI sbírku asi 35 miliónů inkoustových otisků prstů. Tyto bylo třeba skladovat a distribuovat mezi všechna oddělení země. S nástupem technologie bylo logické převést tyto obrázky do digitální podoby a umožnit tak počítačové porovnávání otisků. Vyvstal ovšem problém: jak uložit tyto obrázky v dostatečné kvalitě když jejich počet narostl přes 500 milionů? Použít nějakou existující kompresi nebylo možné: bezztrátové komprese umí kompresní poměr nejvýše 3 ku 1 a FBI potřebovalo poměr přes 10 ku jedné. Použití v té době populárního JPEG zase nebylo možné kvůli artefaktům. Do praxe se tedy zavedla metoda waveletové transformace. Ta sestává podobně jako JPEG z transformace obrazu do frekvencí a následně quantizace a uložení výstupu jako sekvence. Druhé dva kroky probíhají velmi podobně, jako bylo popsáno výše s tím rozdílem že quantizace má o něco složitější matici která je dokonce v některých případech schopná se přizpůsobovat podobně jako například při výpočtech proudění vzduchu kolem křídel letadel, ukládání komprimovaného otisku je také sofistikovanější.

Princip waveletová transformace spočívá v použití dvou filtrů, to je zásadní rozdíl oproti transformacím používající Fourierovy transformace kde jsou data více méně přenásobena nějakou maticí. Jeden z filtrů je tzv. low-pass, druhý high-pass, data projdou oběma filtry, výsledkem jsou potom dvě matice – jedna reprezentující detaily obrazu, druhá reprezentující plochy a pozadí. Autor algoritmu si vybírá, jaké filtry použije (oproti DCT která je stále stejná) a lze je opět použít (rekurzivně) na již přefiltrovaná data, tak vzniká stromová struktura.

Co je wavelet?: funkce $\psi(t)$se nazývá wavelet pokud existuje odpovídající funkce $\phi(t)$nazvaná škálovací funkce a 4 posloupnosti reálných čísel g_k,h_k,p_k a q_k, splňujících:

$$\phi(t) = \sum_k p_k \phi(2t-k), a\ \psi(t) = \sum_k q_k \phi(2t-k)$$

$$\phi(2t-n) = \frac{1}{2}\sum_k [g_{2k-n}\phi(t-k) + h_{2k-n}\psi(t-k)]$$

Dále, aby ϕ a ψtvořily uspokojivý systém je dále požadováno:

$$\sum_k g_k = \sum_k p_k = 2\ a\ \sum_k h_k = \sum_k q_k = 0$$

Máme-li wavelet definovaný pomocí g_k,h_k,p_k a q_k, waveletová transformace posloupnosti x_k délky N na posloupnost y_k stejné délky má tvar: prvních N/2 členů (u_k) nazvaných low-frequency podposloupnost a druhých N/2 členů (v_k) nazvaných high-frequency podposloupnost:

$$u_k = \sum_n \frac{g_{2k-n}}{\sqrt{2}} x_n\ a\ \ v_k = \sum_n \frac{h_{2k-n}}{\sqrt{2}} x_n$$

Filtry jsou v tomto případě $\frac{g_k}{\sqrt{2}}$ a $\frac{h_k}{\sqrt{2}}$.

Během vývoje algoritmu byly pokládány další nároky na „dobrý“ wavelet a bylo vyselektováno 4297 vyhovujících waveletů, ty byly dále podrobeny rozsáhlým testům a bylo zjištěno, že pouze 18 z nich je vhodných ke kompresy, jeden z těchto 18 je používán FBI ke kompresy otisků prstů. Na závěr ukázka waveletové komprese a porovnání s JPEG kompresí:

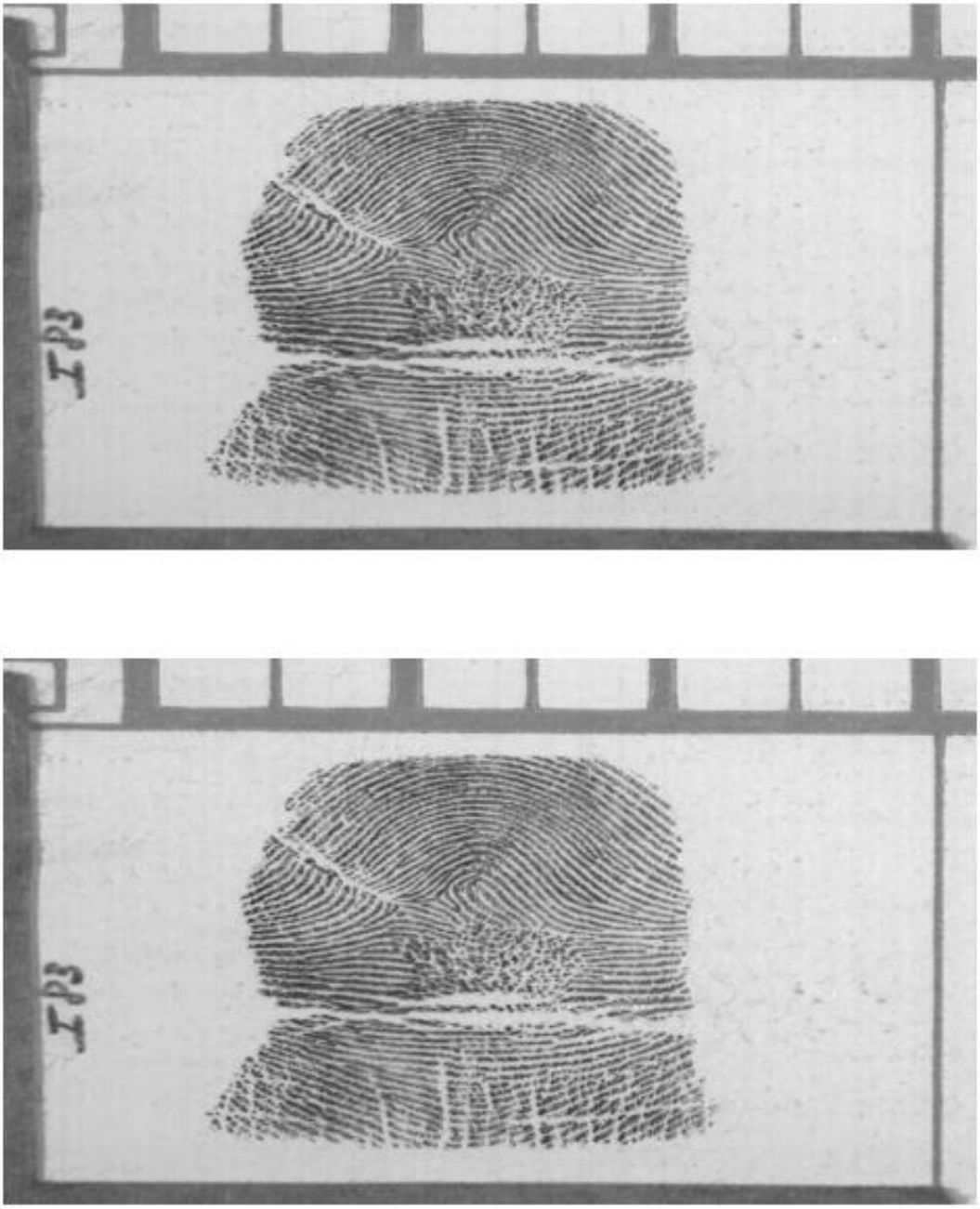


Figure 10: **An Original Fingerprint Image (above) and its WSQ-Recosntructed at 15:1 (below)**

Původní obrázek a Komprimovaný obrázek poměrem 15:1

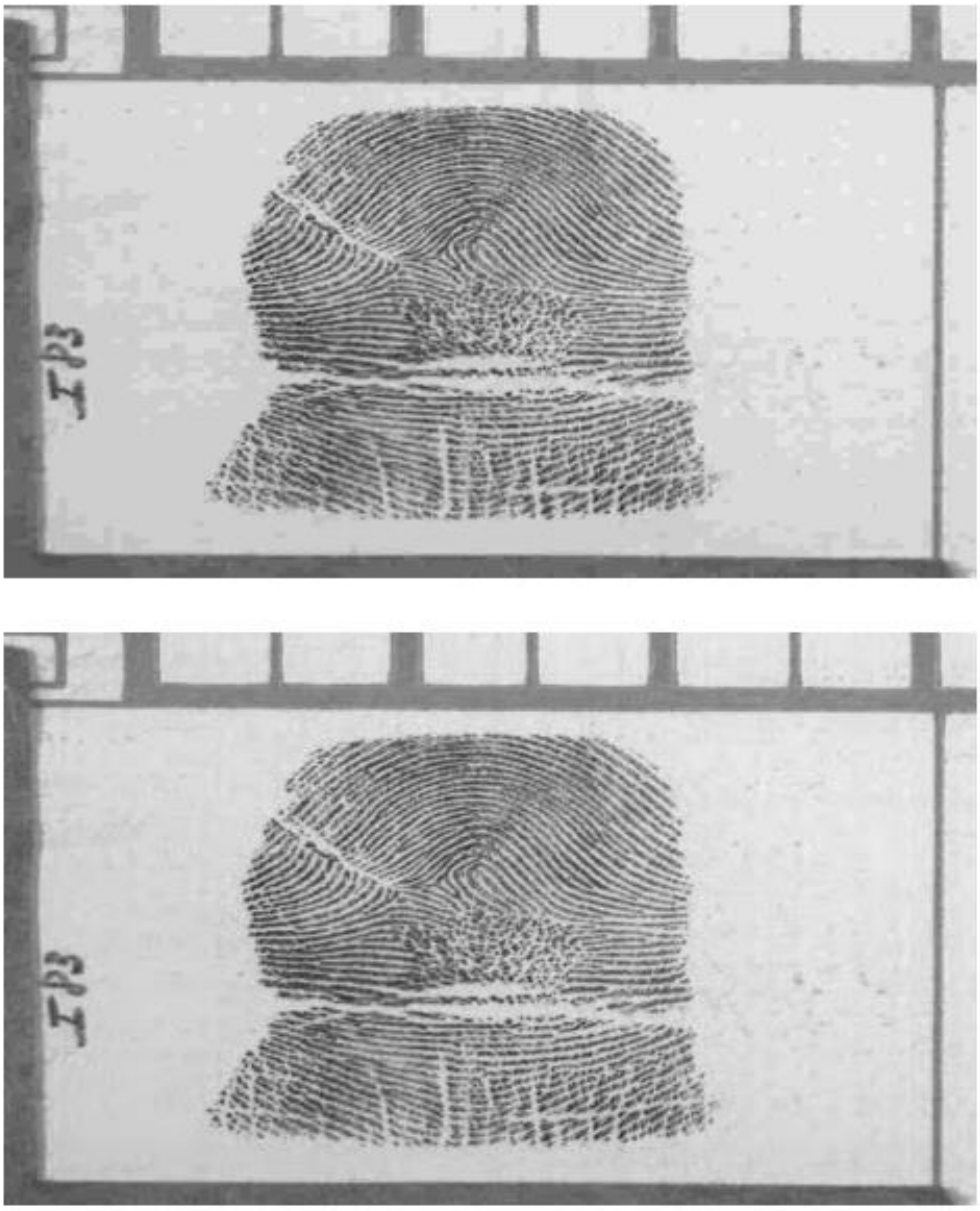


Figure 14: **JPEG at 30:1 (above) and WSQ at 30:1 (below)**

Porovnání JPEG a waveletové transformace pro kompresní poměr 1:30, u JPEG obrázku (nahoře) lze vidět v levé části artefakty (kostičkování) zatímco waveletová transformace je na první pohled kvalitnější.

Zdroje:

Analýza metod pro maticové výpočty, J. D. Tebbens, I. Hnětynková, M. Plešinger, Z. Strakoš, P. Tichý, 2012, matfyzpress

Singulární rozklad matice a jeho použití, Diplomová práce, Eva Vodrážková, 2012, Masarykova universita, přírodovědecká fakulta, ústav matematiky a statistiky

http://demonstrations.wolfram.com/ImageCompressionViaTheSingularValueDecomposition/

http://inst.eecs.berkeley.edu/~ee127a/book/login/l_svd_apps_image.html

http://fourier.eng.hmc.edu/e161/lectures/svdcompression.html

http://volunteerlecturerprogram.com/wp-content/uploads/2013/01/vuthythesis.pdf

http://en.wikipedia.org/wiki/JPEG

http://www.shmo.de/mlab/fourier.html

http://homepages.inf.ed.ac.uk/rbf/HIPR2/fourier.htm

http://www.seas.gwu.edu/~ayoussef/papers/FingerPrintWSQ-chapter.pdf