

# Metody předzpracování obrazu – barevné, jasové a geometrické

Strojové vidění a zpracování obrazu (BI-SVZ)

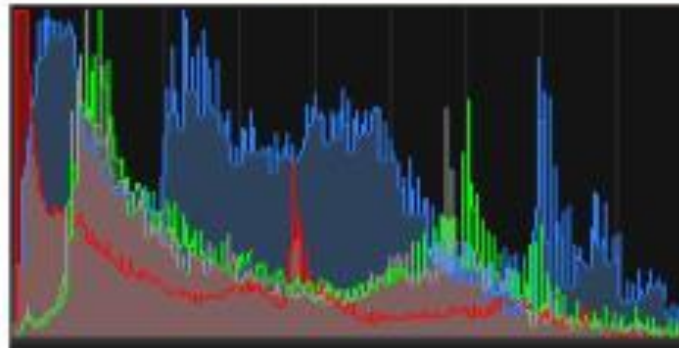
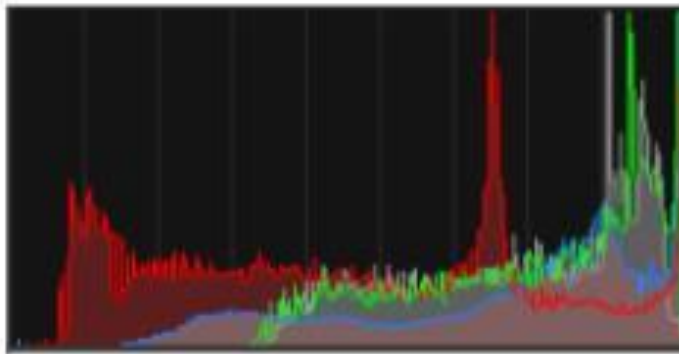
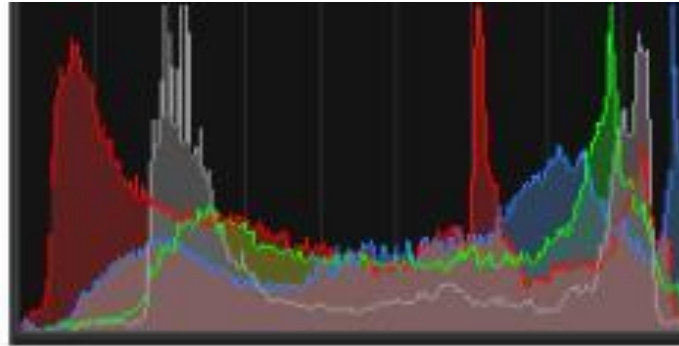
# Motivace předzpracování obrazu

- Nevhodná volba vyvážení bílé
- Chybné nastavení expozice
- Požadavky na jiný barevný prostor
- Odstranění šumu, zaostření snímku
- Zisk relevantních regionů
- Geometrické zkreslení znemožňující aplikaci některých algoritmů (OCR)
- Komprese dat

# Typy předzpracování obrazu

- Barevné a jasové transformace
  - Úprava jasu, kontrastu, ...
  - Ekvalizace histogramu
  - Zvýraznění určitých charakteristik obrazu
  - Prahování
  - Hranové detekce
  - Filtrace a vyhlazování
  - ...
- Geometrické transformace
  - Odstranění soudkovitosti
  - Euklidovské, afinní, projektivní, ...
- Frekvenční transformace

# Barevné a jasové transformace



# Převod barevného RGB snímku na černobílý

- Průměrovací metoda

- $$I = \frac{R+G+B}{3}$$

- Metoda váhování

- $$I = 0,3R + 0,59G + 0,11B$$

- V průměrovací metodě bereme 33 % hodnotu z každého RGB kanálu, ve skutečnosti však všechny barvy nepřispívají stejným dílem (fyzikální vlastnosti, snímač, apod.)



Originál



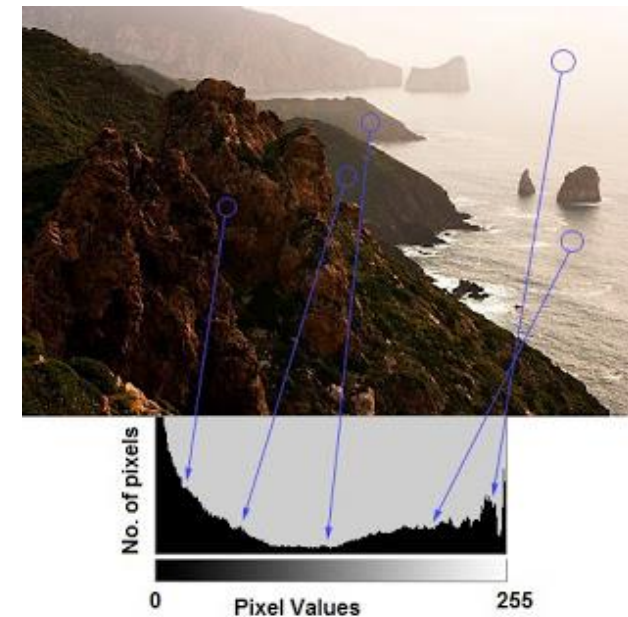
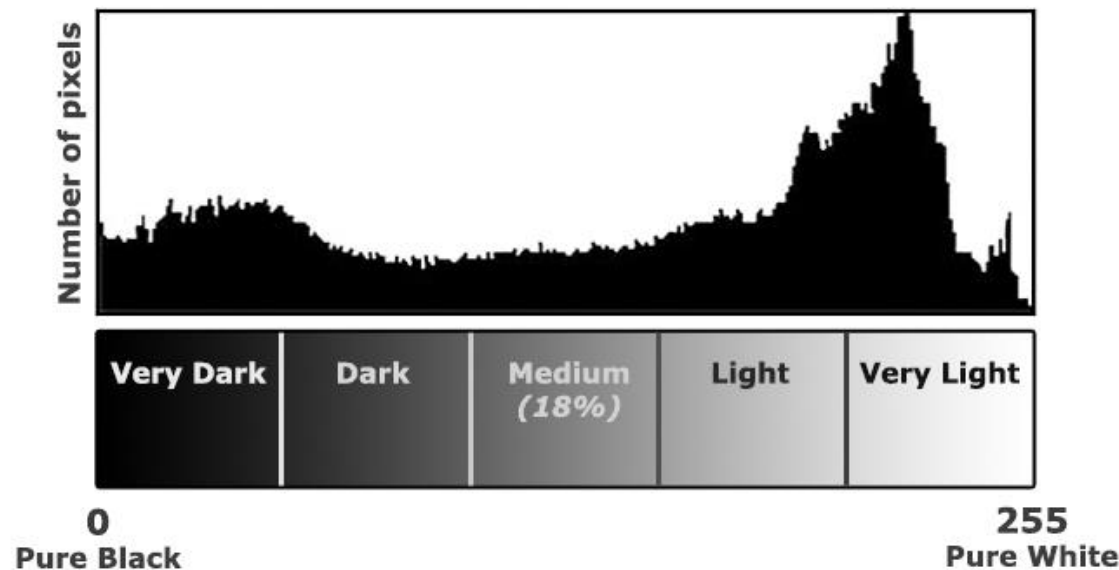
Převod průměrováním



Převod váhováním

# Histogram

- Grafové znázornění distribuce jasových hodnot pixelů
- Dokáže prozradit, zda je snímek vhodně exponován, zda není světlo příliš mdlé nebo ostré, případně jaké úpravy na snímek aplikovat
- Osa Y vyjadřuje četnost v daném intervalu.
- Kromě jasů existuje i pro jednotlivé RGB kanály.
- Výpočet v OpenCV - [`cv2.calcHist\(images, channels, mask, histSize, ranges\[, hist\[, accumulate\]\]\)`](#)



# Ukázky histogramů



**Dominující tmavé odstíny  
tzv. Low-key metoda (tmavá tonalita)**



**Dominující světlé odstíny  
tzv. High-key metoda (světlá tonalita)**



**Fotografie s vysokým  
kontrastem**



**Fotografie s nízkým  
kontrastem**

# Ekvalizace histogramu

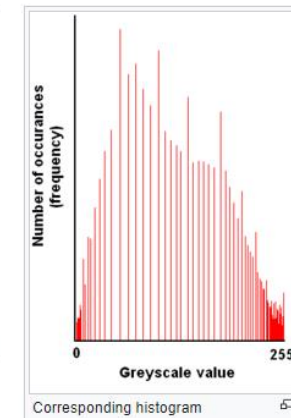
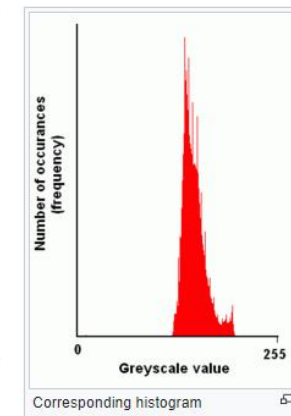
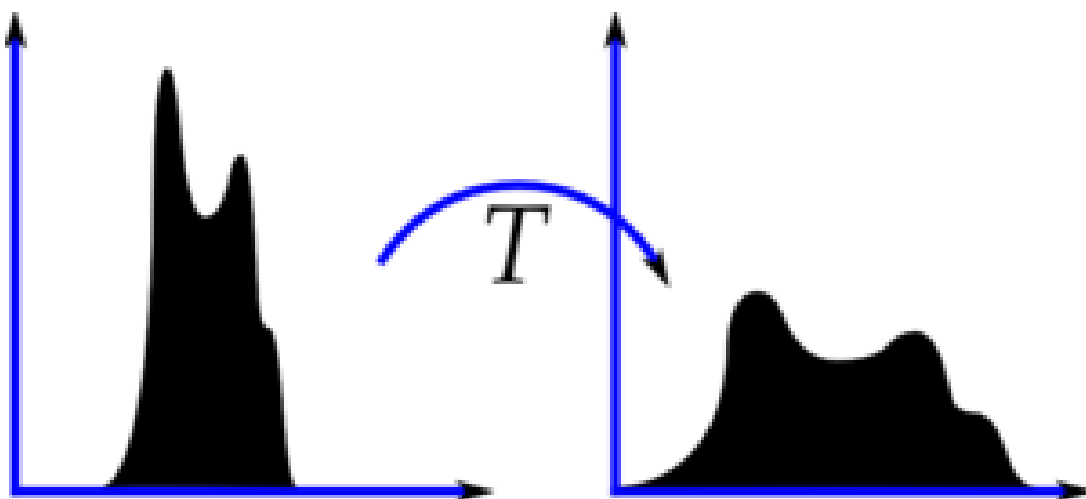
- Zajišťuje snazší interpretaci vizualizovaného obrazu pomocí zvýšení lokálního kontrastu
- Užitečné pro obrazy, které jsou příliš tmavé, příliš světlé, nebo nekонтastní
- V ekvalizovaném histogramu jsou jasové úrovně zastoupeny zhruba stejně četně
- Jednoduchá transformace na výpočet, a zároveň invertibilní
- Nevýhodou je zvýrazněný šum v obraze
- Implementace v OpenCV - [cv2.equalizehist\(hist\)](#)



# Ekvalizace histogramu

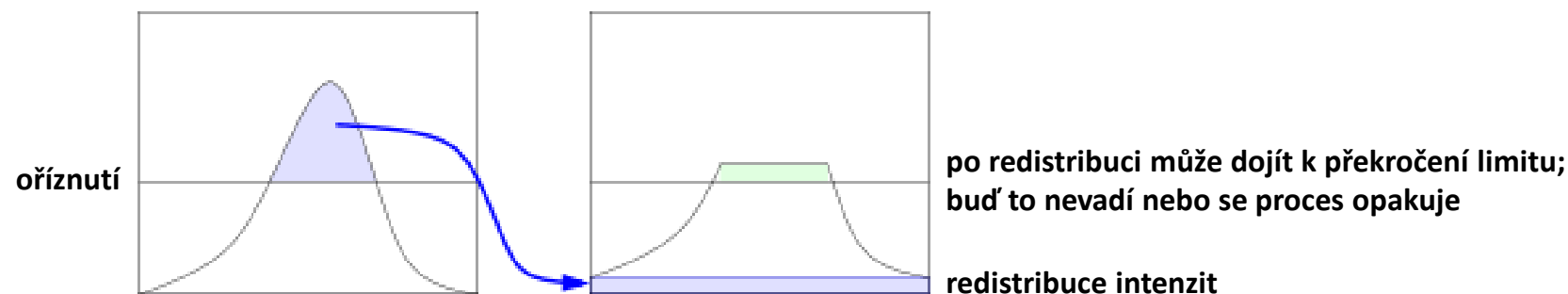
- Využití:

- Face recognition, kde usilujeme o co nejpodobnější jasové podmínky pro celý dataset
- Rentgenové snímky
- Snímky zaznamenané termokamerou
- Chybně exponované snímky

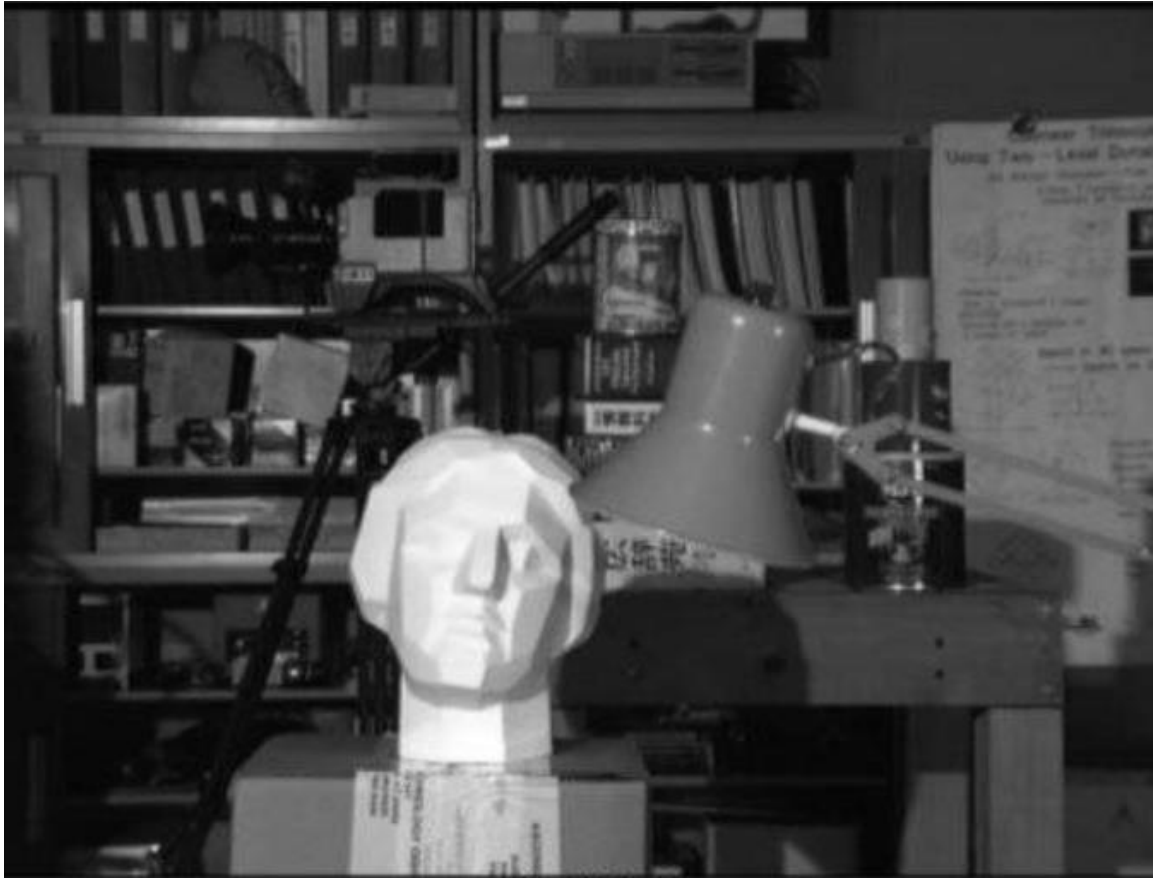


# CLAHE - Contrast Limited Adaptive Histogram Equalization

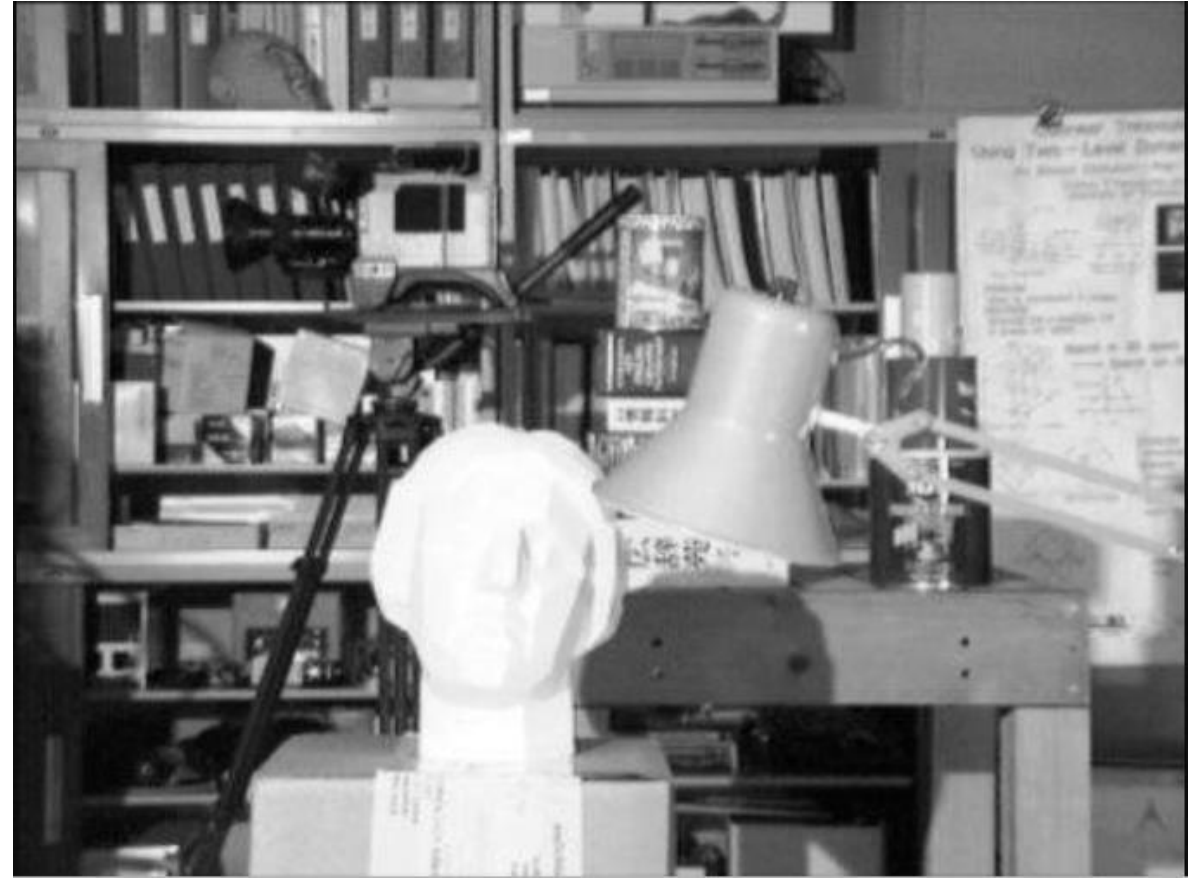
- Klasická verze ekvalizace uvažuje pouze globální transformaci
- AHE (Adaptive Histogram Equalization) využívá k výpočtu klouzavé okénko, kde pro každý region vypočte lokální histogram
- Následně se provede ekvalizace nad tímto regionem
- AHE může vést k tomu, že dojde k přílišnému zvýraznění šumu, zejména nekontrastního regionu -> “chytrý” clipping (Contrast Limited AHE = CLAHE)
- Implementace v OpenCV pomocí [cv2.createCLAHE\(\)](#)



# CLAHE - Contrast Limited Adaptive Histogram Equalization

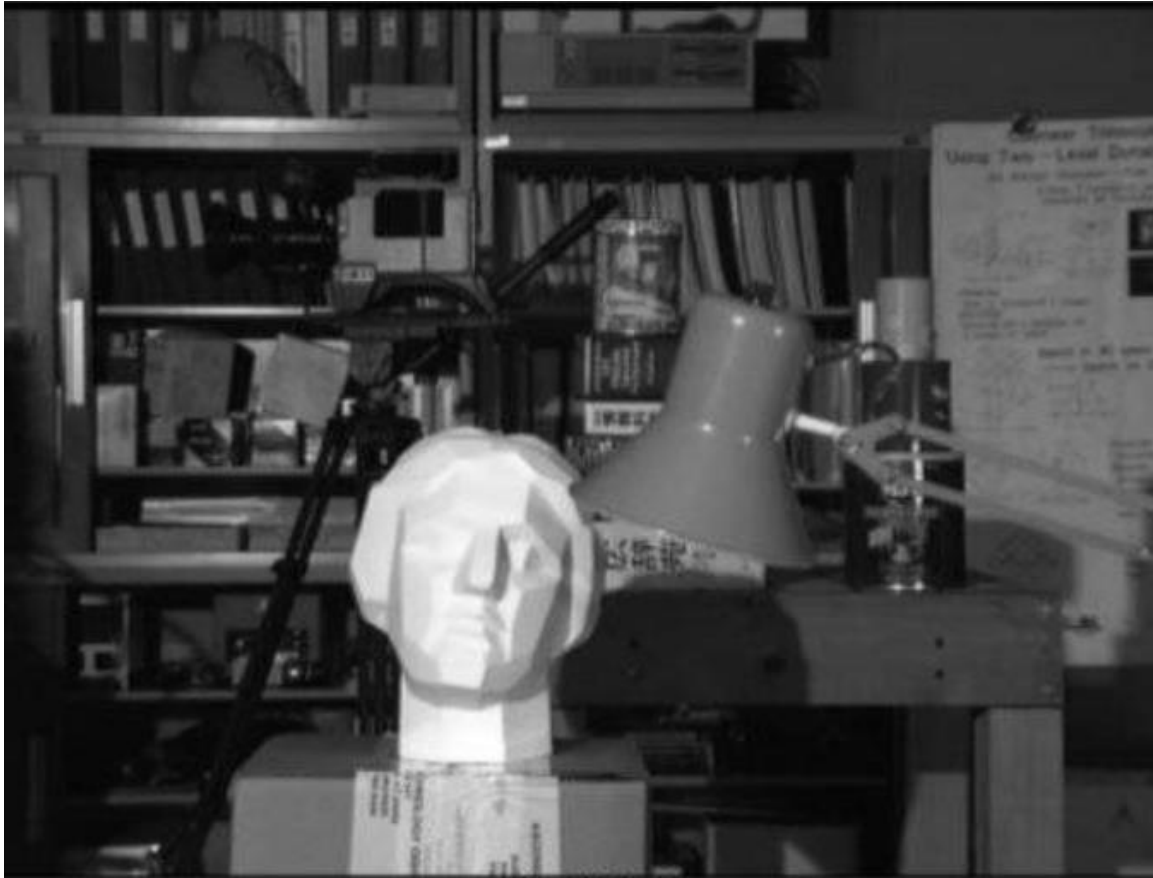


Original

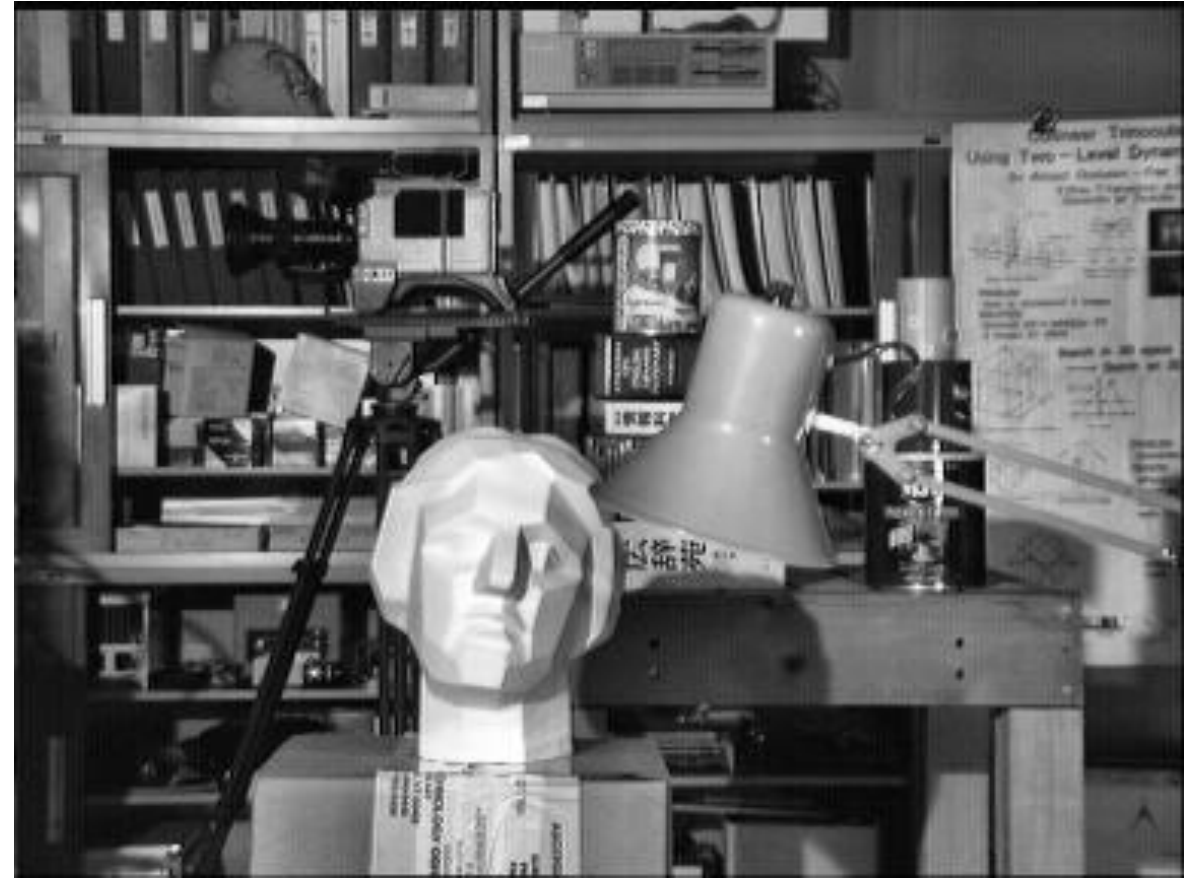


Snímek po ekvalizaci histogramu

# CLAHE - Contrast Limited Adaptive Histogram Equalization



Original



Snímek po aplikaci CLAHE

# Porovnávání histogramů

- Velmi naivní způsob k zjištění podobnosti dvou obrázků
- Označme histogramy dvou různých obrázků  $H_1$  a  $H_2$
- K porovnání  $H_1$  a  $H_2$  zavedme podobnostní metriku  $d(H_1, H_2)$

## Příklady metrik:

- Korelační
  - Čím větší, tím podobnější – maximální hodnota je 1

- $$d(H_1, H_2) = \frac{\sum_I (H_1(I) - \bar{H}_1)(H_2(I) - \bar{H}_2)}{\sqrt{\sum_I (H_1(I) - \bar{H}_1)^2 \sum_I (H_2(I) - \bar{H}_2)^2}}$$

- kde  $\bar{H}_k = \frac{1}{N} \sum_J H_k(J)$

# Porovnávání histogramů – příklady metrik

- Chí-kvadrát

- Výpočet chí-kvadrát vzdálenosti – čím menší, tím podobnější

- $$d(H_1, H_2) = \sum_I \frac{(H_1(I) - H_2(I))^2}{H_1(I)}$$

- Průnik

- Výpočet průniku - čím větší, tím podobnější

- $$d(H_1, H_2) = \sum_I \min(H_1(I), H_2(I))$$

- Bhattacharyyova vzdálenost (někdy také nazývána Hellingerova)

- Nabývá hodnot z intervalu  $< 0, 1 >$  - čím menší, tím podobnější

- $$d(H_1, H_2) = \sqrt{1 - \frac{1}{\sqrt{H_1 H_2} N^2} \sum_I \sqrt{H_1(I) \cdot H_2(I)}}$$

# Porovnávání histogramů – implementace

- Porovnávání dvou histogramů v OpenCV - [cv2.compareHist\(H1, H2, metric\)](#)

Kde hodnota *metric* může být:

- cv2.HISTCMP\_CORREL – korelační
- cv2.HISTCMP\_CHISQR – chí-kvadrát vzdálenost
- cv2.HISTCMP\_CHISQR\_ALT – alternativní chí-kvadrát vzdálenost
- cv2.HISTCMP\_INTERSECT – průnik
- cv2.HISTCMP\_BHATTACHARYYA – Bhattacharyya vzdálenost měřící překryv histogramů
- cv2.HISTCMP\_HELLINGER – synonymum pro Bhattacharyya
- cv2.HISTCMP\_KL\_DIV - Kullback-Leibler divergence

(Pozor, v každé verzi OpenCV trochu jiné názvy metrik)



# Porovnávání histogramů – příklad na obrázku





# Úprava jasu

- Dále předpokládáme 2D obrazový snímek reprezentovaný pomocí matice, funkci  $f(i, j)$ , která vrátí hodnotu/vektor pixelu v řádku  $i$  a sloupci  $j$  a funkci  $g(i, j)$ , která vrací hodnotu/vektor pixelu po transformaci
- Triviální příklad zvýšení jasu můžeme provést přičtením konstanty  $\beta > 0$  ke každému pixelu (naopak snížení jasu přičtením  $\beta < 0$ )
  - $g(i, j) = f(i, j) + \beta$

# Úprava jasu - příklad

- Originální obrázek:

12	23	84	122
123	34	92	200
23	45	29	73

- Zvýšení jasu o 60:

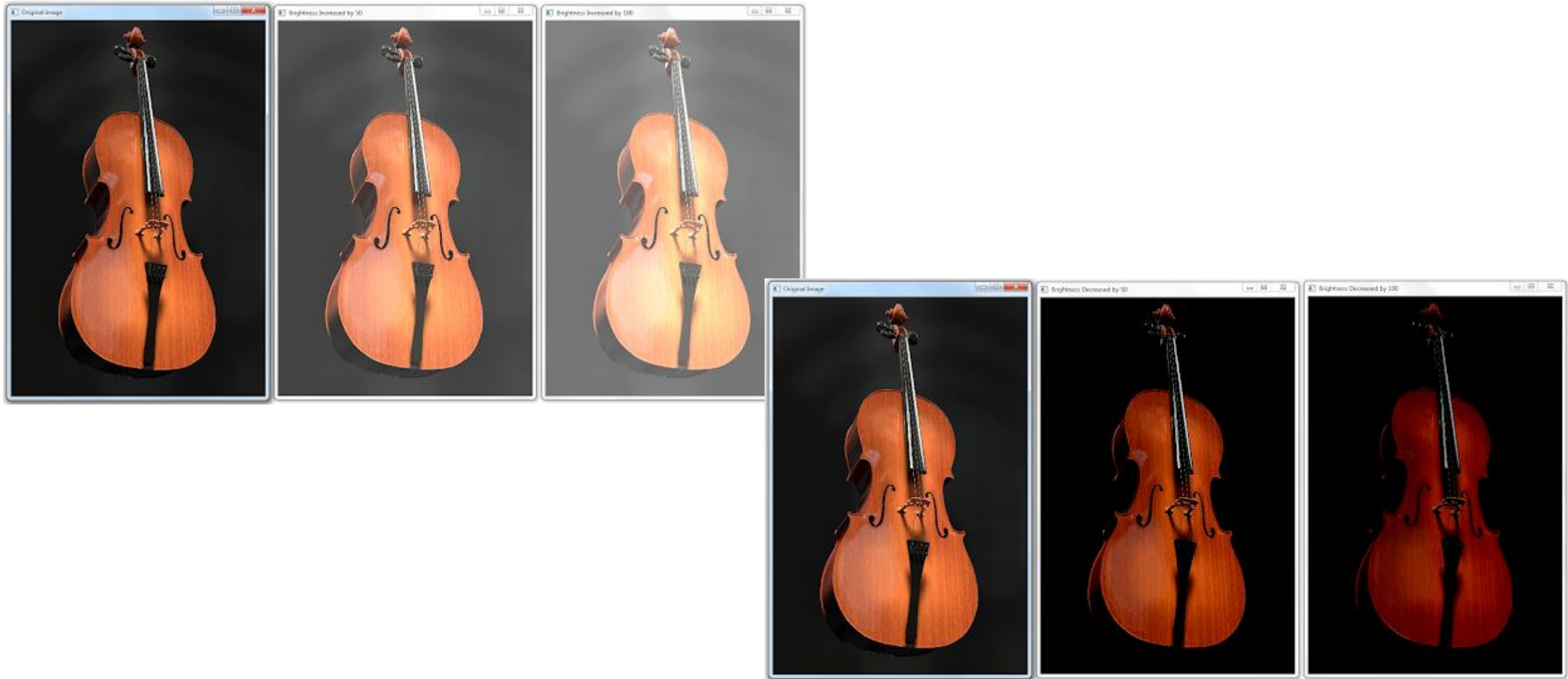
12 + 60	23 + 60	84 + 60	122 + 60
123 + 60	34 + 60	92 + 60	200 + 60
23 + 60	45 + 60	29 + 60	73 + 60

 = 

72	83	144	182
183	94	152	255
83	105	89	133

- Při přesažení hodnoty 255 ztrácíme v 8 bitovém obraze informace

# Úprava jasu - příklad



# Úprava kontrastu

- Změna kontrastu  $g(i, j) = \alpha * f(i, j)$ 
  - Snížení pro  $0 < \alpha < 1$
  - Zvýšení pro  $\alpha > 1$
- Originální obrázek:

144	245	132	54
10	62	81	84
99	106	29	7

- Zvýšení kontrastu o faktor 2

$144 * 2$	$245 * 2$	$132 * 2$	$54 * 2$
$10 * 2$	$62 * 2$	$81 * 2$	$84 * 2$
$99 * 2$	$106 * 2$	$29 * 2$	$7 * 2$

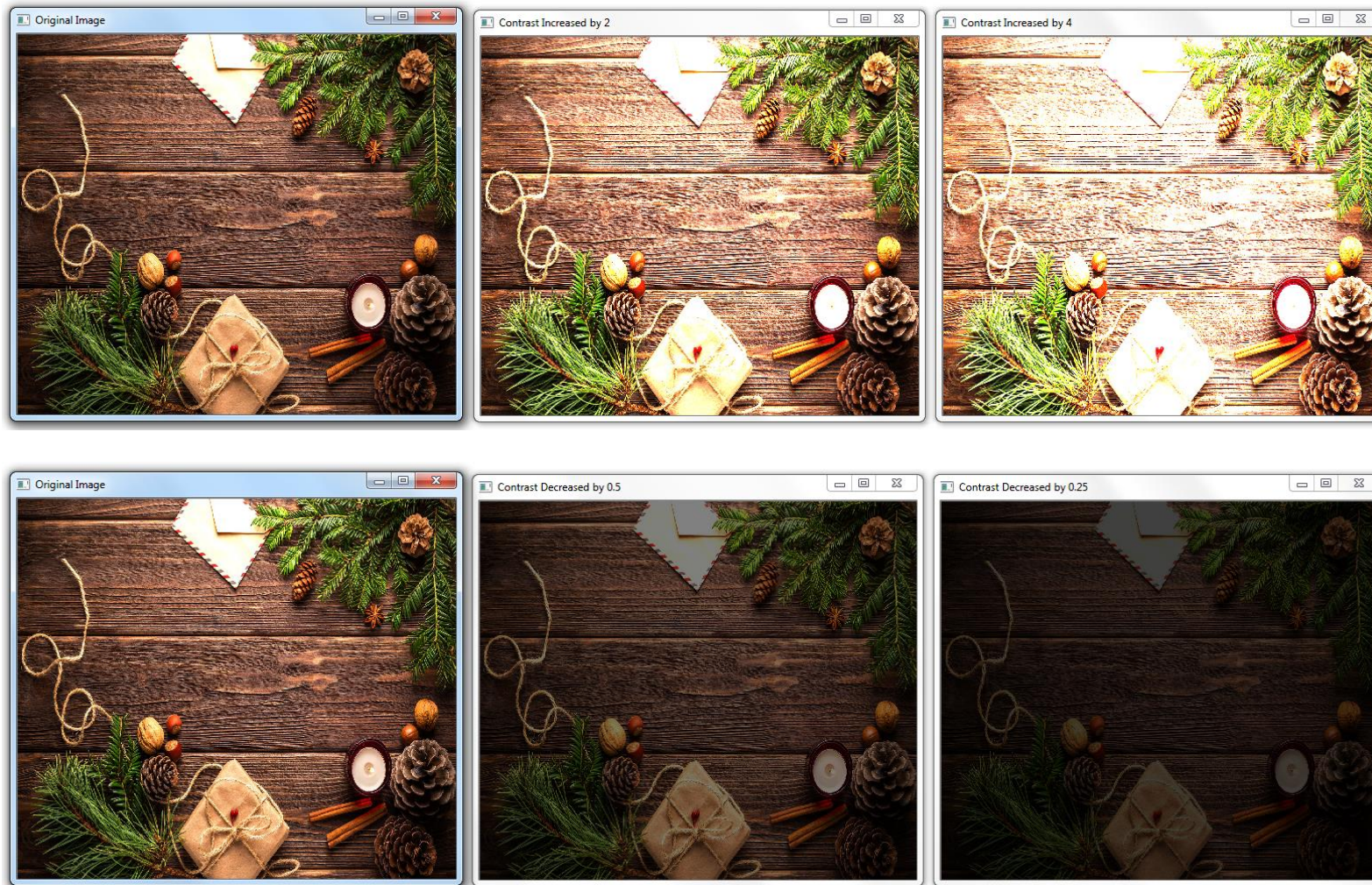
 $=$ 

255	255	255	108
20	124	162	168
198	212	58	14

- Při přesažení hodnoty 255 ztrácíme v 8 bitovém obraze informace



# Úprava kontrastu - příklad



# Změna jasů a kontrastu dohromady

- Nejčastěji se však setkáme s formulací
  - $g(i, j) = \alpha * f(i, j) + \beta$
  - viz dokumentace v [OpenCV](#)

Je jasová a kontrastní transformace invertibilní?



# Změna jasu a kontrastu dohromady

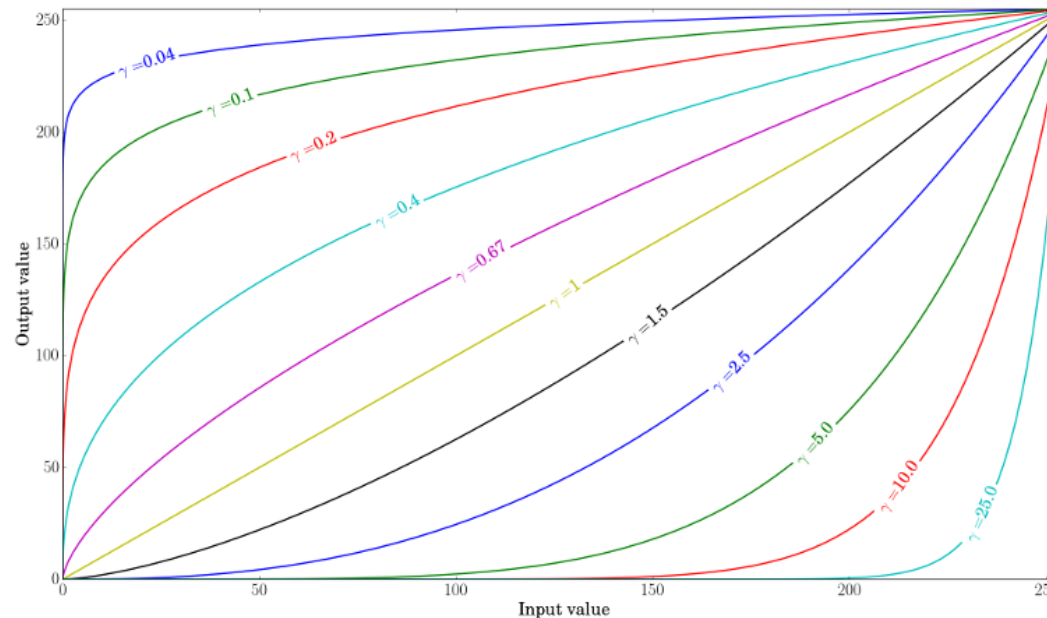
Nemusí být, protože mohlo dojít k přetečení (clippingu) a tedy ke ztrátě informace



Ukázka přidání jasu a zvýšení kontrastu:  $\alpha = 1.3$ ;  $\beta = 40$

# Gamma korekce

- Je nelineární transformace všech pixelů, která se snaží dát stínům a světlům více prostoru
  - $g(i, j) = \left(\frac{f(i, j)}{255}\right)^\gamma * 255$
  - Pro  $\gamma < 1$  zesvětlení stínů - posun histogramu doprava
  - Pro  $\gamma > 1$  ztmavení světel - posun histogramu doleva





# Gamma korekce



Ukázka gamma korekce pro  $\gamma = 0.4$

# Úprava jasu vs gamma korekce – příklad histogramu

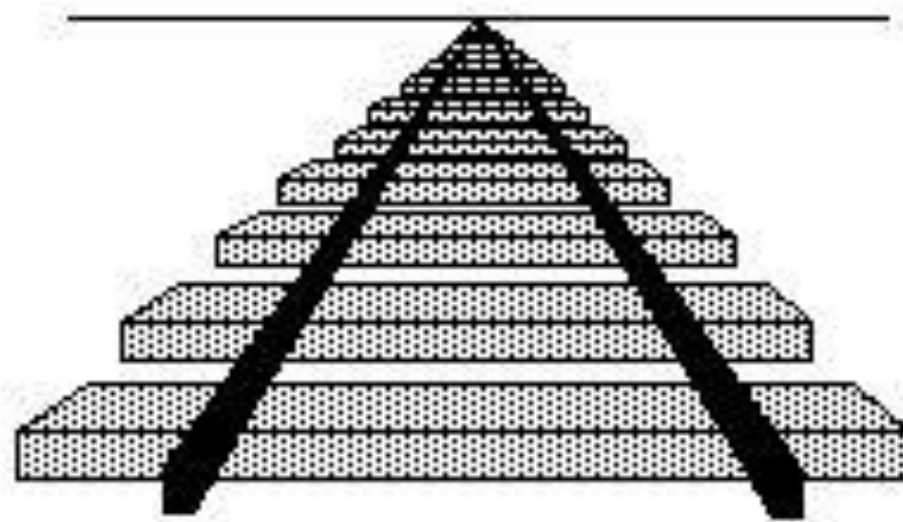
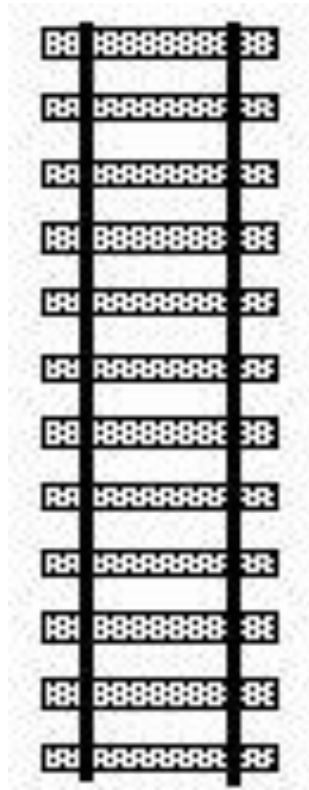


Zesvětlení pomocí zvýšení jasu  
(přičtení  $\beta$ )

Histogram originálního snímku

Zesvětlení pomocí gamma korekce

# Geometrické transformace



# Typy geometrických transformací

- Nejpoužívanější transformace v 2D rovině:

- Posunutí
- Euklidovská (lineární)
- Podobnostní
- Afinní
- Projektivní

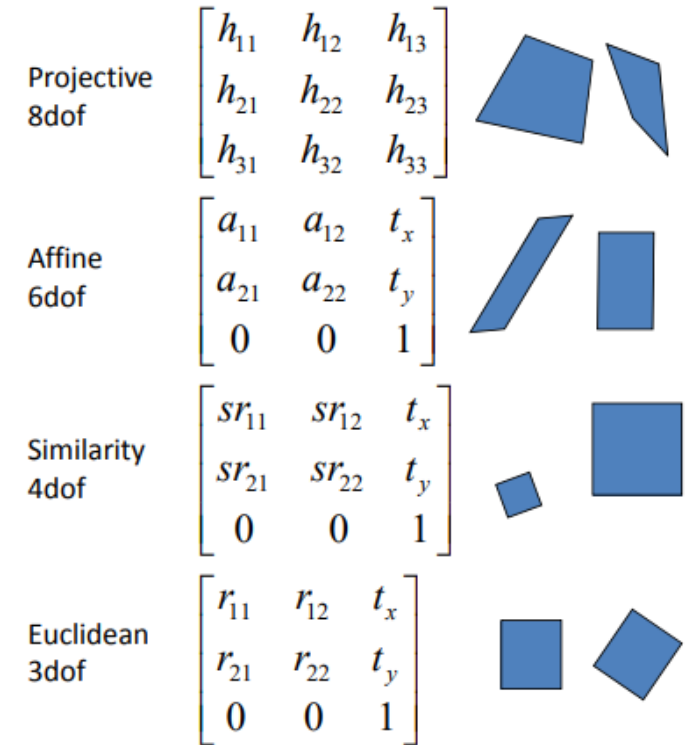
- 3D transformace jsou obdobné

- využívají matice 4x4

- Hierarchie:

- Euklidovské  $\subset$  Podobnostní  $\subset$  Afinní  $\subset$  Projektivní

A square transforms to:



# Geometrické transformace – posunutí

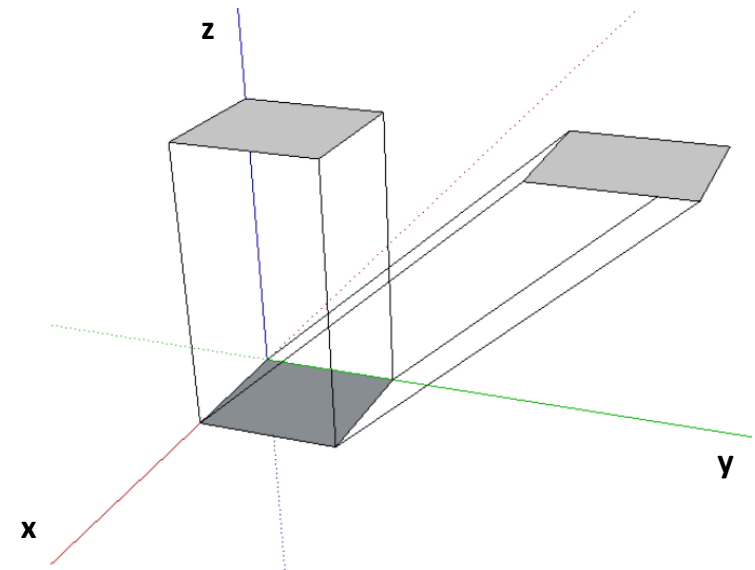
- Příklad ve 2D

- $(u_x, u_y) = (v_x, v_y) + (t_x, t_y) \rightarrow u = v + t$

- Maticově

- Výhodné, neboť v homogenních souřadnicích se jedná o lineární operaci
  - Díky tomu se 2D posun vyjádří pomocí operace 3D zkosení

- $$\begin{pmatrix} u_x \\ u_y \\ 1 \end{pmatrix} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{pmatrix} v_x \\ v_y \\ 1 \end{pmatrix}$$



# Geometrické transformace – Euklidovská

- Rotace a posunutí
- Zachovává velikosti úhlu, poměry ploch a vzdálenosti mezi body (izometrické zobrazení) – proto ten název

- Příklad ve 2D

- $(u_x, u_y) = (v_x \cos \theta - v_y \sin \theta + t_x, v_x \sin \theta + v_y \cos \theta + t_y)$
- Kde  $\theta$  je úhel otočení od vodorovné osy (x) kolem počátku souřadného systému

- Maticově

- $u = T \cdot R \cdot v$

- $$\begin{pmatrix} u_x \\ u_y \\ 1 \end{pmatrix} = \begin{bmatrix} \cos \theta & -\sin \theta & t_x \\ \sin \theta & \cos \theta & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{pmatrix} v_x \\ v_y \\ 1 \end{pmatrix}$$

$$T = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \quad R = \begin{bmatrix} \cos \theta & -\sin \theta & 0 \\ \sin \theta & \cos \theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

- Operace rotace a posunutí ve 3D, viz [Vysvětlení](#)

# Geometrické transformace – podobnostní

- Škálování, rotace a posunutí
- Zachovává velikosti úhlů a poměr vzdálenosti bodů na přímce
- Příklad ve 2D
  - $(u_x, u_y) = (sv_x \cos \theta - sv_y \sin \theta + t_x, sv_x \sin \theta + sv_y \cos \theta + t_y)$

- Maticově

- $u = T \cdot R \cdot S \cdot v$

$$S = \begin{bmatrix} s & 0 & 0 \\ 0 & s & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

- $$\begin{pmatrix} u_x \\ u_y \\ 1 \end{pmatrix} = \begin{bmatrix} s \cdot \cos \theta & s \cdot (-\sin \theta) & t_x \\ s \cdot \sin \theta & s \cdot \cos \theta & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{pmatrix} v_x \\ v_y \\ 1 \end{pmatrix}$$

# Geometrické transformace – afinní

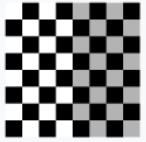
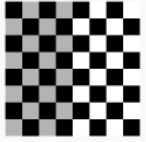
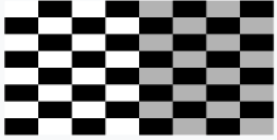


- Skládá se z kombinace lineárních transformací (škálování, rotace a zkosení) a posunu
- Zachovává [kolinearitu](#), vlastnost rovnoběžnosti, poměr vzdálenosti bodů na přímkách a poměry ploch
- Zobrazení mezi afinními prostory - všechny Euklidovské prostory jsou afinní, ale ne všechny afinní jsou Euklidovské.
- Transformace nemusí nutně zachovávat úhly, vzdálenosti a souřadnice počátku (nulový bod)
- Každá lineární transformace je afinní, ale ne každá afinní transformace je lineární (díky nezachovávání souřadnic počátku)



# Geometrické transformace – afinní

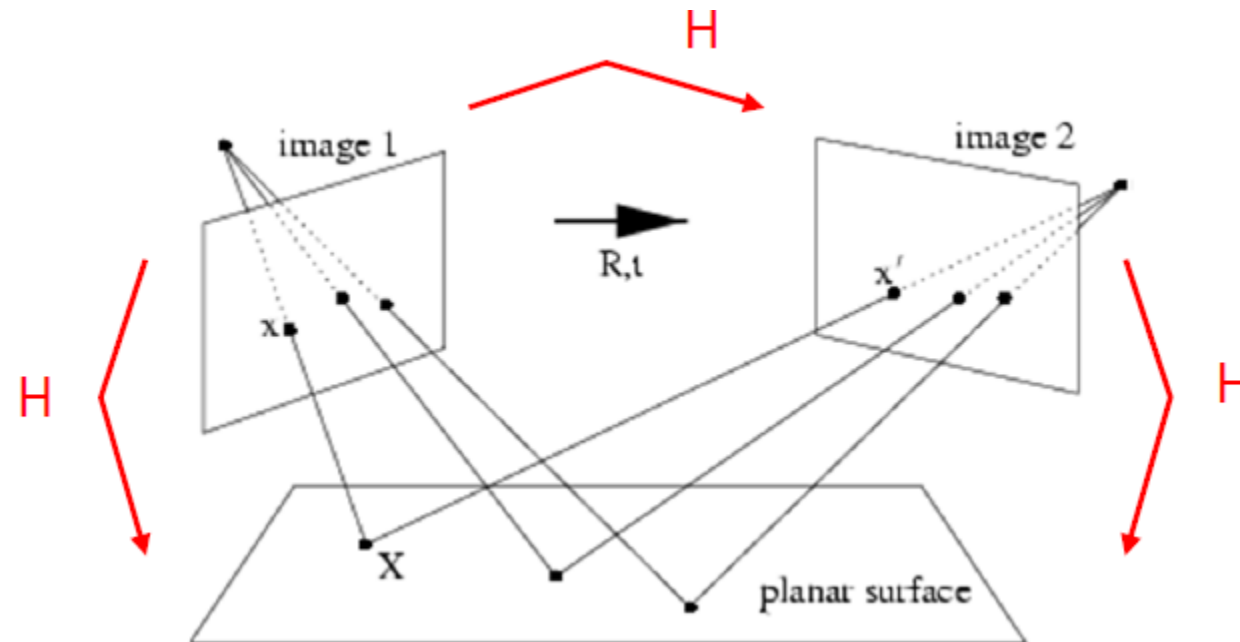
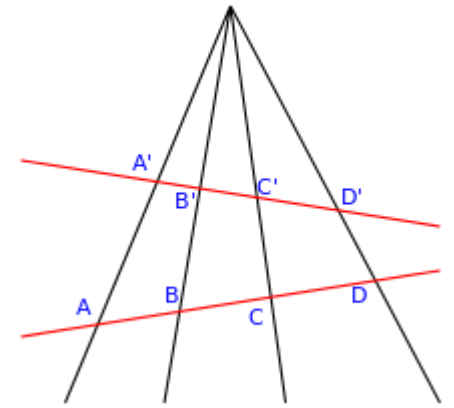
- Není natolik silný nástroj, aby transformoval čtverec na libovolný čtyřúhelník – k tomu se využívá projektivní transformace
- Nejčastější využití je v počítačové grafice
- Pozor, záleží na pořadí prováděných jednotlivých operací.
- Výslednou transformační matici získáme pronásobením dílčích matic.
- Násobení matic není komutativní.
- Maticově ve 2D
  - $u = A \cdot v$

$$\begin{pmatrix} u_x \\ u_y \\ 1 \end{pmatrix} = \begin{bmatrix} a_{0,0} & a_{0,1} & a_{0,2} \\ a_{1,0} & a_{1,1} & a_{1,2} \\ 0 & 0 & 1 \end{bmatrix} \begin{pmatrix} v_x \\ v_y \\ 1 \end{pmatrix}$$

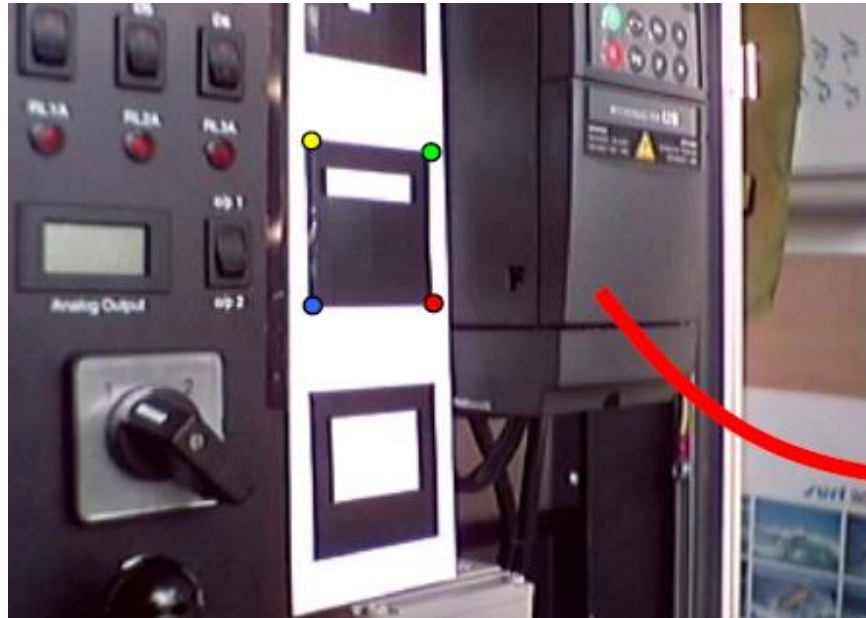
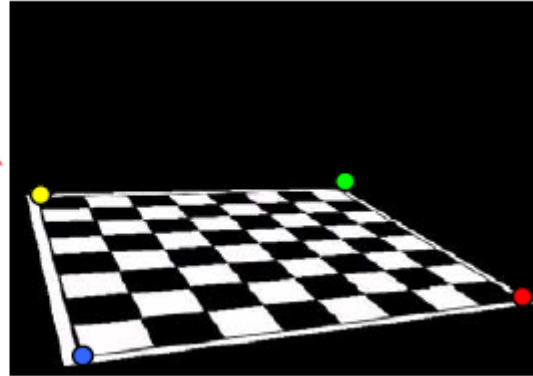
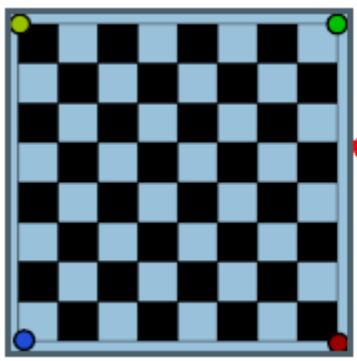
Transformation name	Affine matrix	Example
<b>Identity</b> (transform to original image)	$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$	
<b>Reflection</b>	$\begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$	
<b>Scale</b>	$\begin{bmatrix} c_x = 2 & 0 & 0 \\ 0 & c_y = 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$	
<b>Rotate</b>	$\begin{bmatrix} \cos(\theta) & \sin(\theta) & 0 \\ -\sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix}$	 where $\theta = \frac{\pi}{6} = 30^\circ$
<b>Shear</b>	$\begin{bmatrix} 1 & c_x = 0.5 & 0 \\ c_y = 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$	

# Geometrické transformace – projektivní

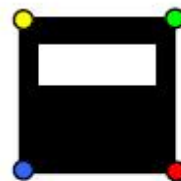
- Zachovává [dvojpoměr](#) a [kolinearitu](#)
- Vyžaduje homogenní souřadnice, neboť je měnící
- Body se souřadnicemi v nekonečnu dokáže převést na konečné a naopak



# 2D projektivní transformace (homography)



lícovací body



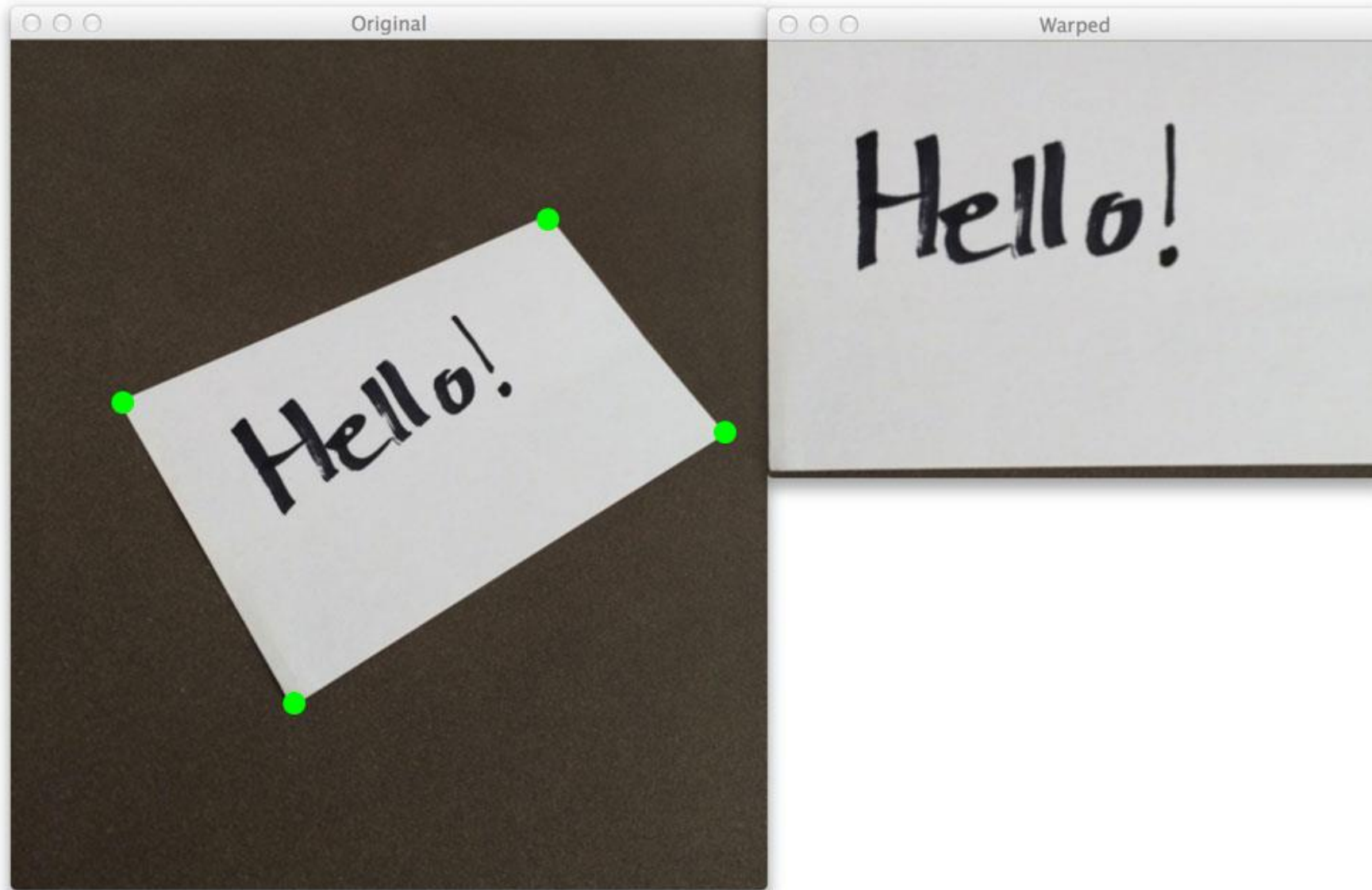
# 2D projektivní transformace (homography)





# 2D projektivní transformace (homography)

- [Příklad](#)

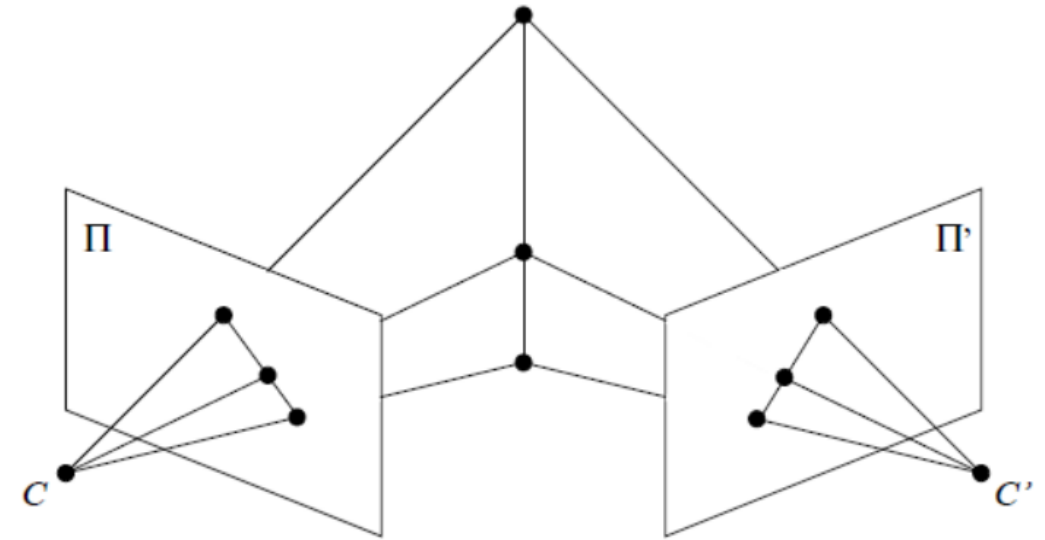


# 2D projektivní transformace (homography)

- Transformace mezi projektivními rovinami
- K výpočtu je nutné znát minimálně 4 bodové korespondence
- Následně se aplikuje algoritmus [DLT](#), tím získáme matici  $H$
- V OpenCV – [cv2.findHomography\(\)](#) a [cv2.warpPerspective\(\)](#)

$$\begin{bmatrix} \rho'_i x'_i \\ \rho'_i y'_i \\ \rho'_i \end{bmatrix} = \tilde{\mathbf{x}}'_i = \mathbf{H} \tilde{\mathbf{x}}_i = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix}$$

|  
image  
point|  
homography|  
image  
point



- Afinní transformace je speciálním případem když:

$$h_{31} = h_{32} = 0, h_{33} = 1$$

[Demo projektivní transformace](#)

# Zdroje

- <https://www.opencv-srf.com>
- <https://www.alza.cz/slovník/histogram>
- <http://www.cambridgeincolour.com/tutorials/histograms1.htm>
- <https://www.pyimagesearch.com/2014/07/14/3-ways-compare-histograms-using-opencv-python/>