



# Robotika: Základy vidění

Vladimír Petřík

[vladimir.petrik@cvut.cz](mailto:vladimir.petrik@cvut.cz)

23.10.2023

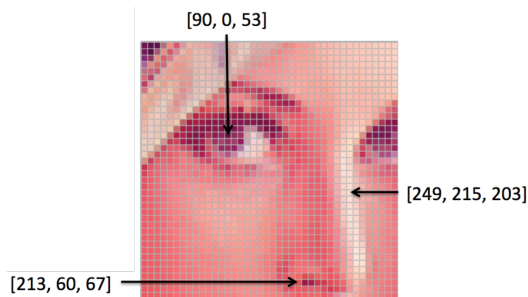
# Co je to obrázek?

- ▶ Kamera připojená k počítači vytváří obrázky
- ▶ Obrázek je pole čísel<sup>1</sup>



187	189	174	149	169	162	129	191	172	142	165	166
185	182	149	74	75	62	39	17	170	219	180	164
180	180	52	14	34	6	13	33	48	136	159	181
206	193	6	134	131	111	120	204	145	15	64	180
164	60	130	281	237	239	239	228	227	67	71	201
172	156	207	233	233	214	220	239	238	61	74	206
188	65	178	209	186	216	211	166	190	75	30	169
189	91	189	14	12	160	126	11	21	62	32	160
199	146	191	193	198	227	178	145	182	194	36	190
209	174	198	282	236	231	149	178	238	43	53	204
190	214	154	149	206	187	65	160	79	38	218	241
190	224	147	146	227	210	127	163	36	198	285	224
190	214	173	96	103	143	94	96	2	198	249	218
187	190	238	76	1	81	47	5	4	237	286	211
183	202	237	191	6	8	12	108	209	198	143	236
196	206	123	207	177	121	139	209	178	13	94	218

187	182	174	168	160	160	129	191	172	142	165	166
185	182	149	74	75	62	39	17	170	219	180	164
180	180	52	14	34	6	13	33	48	136	159	181
206	193	6	134	131	111	120	204	145	15	64	180
164	60	130	281	237	239	239	228	227	67	71	201
172	156	207	233	233	214	220	239	238	61	74	206
188	65	178	209	186	216	211	166	190	75	30	169
189	91	189	14	12	160	126	11	21	62	32	160
199	146	191	193	198	227	178	145	182	194	36	190
209	174	198	282	236	231	149	178	238	43	53	204
190	214	154	149	206	187	65	160	79	38	218	241
190	224	147	146	227	210	127	163	36	198	285	224
190	214	173	96	103	143	94	96	2	198	249	218
187	196	238	76	1	81	47	5	4	237	286	211
183	202	237	191	6	8	12	108	209	198	143	236
196	206	123	207	177	121	139	209	178	13	94	218



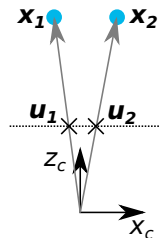
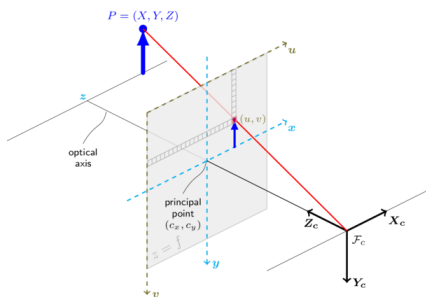
<sup>1</sup>Obrázky jsou z: <https://ai.stanford.edu/~syueung/cvweb/tutorial1.html>



# Jak vzniká obraz?

## ► Perspektivní kamera

- model dírkové kamery/středové promítání<sup>2</sup>
- promítá prostorový bod  $x_c$  do obrazového bodu  $u = (u \ v)^T$  průtnutím
  - obrazové roviny a
  - přímky spojující  $x_c$  se středem projekce
- všechny body na paprsku se promítají do stejného pixelu



<sup>2</sup>[docs.opencv.org](https://docs.opencv.org)



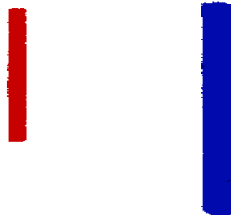
# Projekce

- ▶  $\mathbf{u}_H = K \mathbf{x}_c$ 
  - ▶  $\mathbf{u}_H$  je pixel v homogenních souřadnicích
  - ▶ pokud  $\mathbf{u}_H = (u_H \ v_H \ w_H)^\top$ , pak souřadnice pixelu jsou  $(u_H/w_H \ v_H/w_H)^\top$
  - ▶ alternativně můžeme reprezentovat jako:  $\lambda (u, v, 1)^\top = K \mathbf{x}_c$
- ▶  $K$  je matice kamery
  - ▶  $K = \begin{pmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{pmatrix}$
  - ▶ Co představuje  $\lambda$ ?
    - ▶  $\lambda$  je nenulové reálné číslo
    - ▶ pokud znáte hodnotu  $\lambda$ , můžete vypočítat kartézskou souřadnici  $\mathbf{x} = \lambda K^{-1} \mathbf{u}$   
 $\mathbf{x} = \lambda K^{-1} \mathbf{u}$
    - ▶ v opačném případě lze vypočítat pouze paprsek
  - ▶ jak zjistit  $K$  z bodů?



## Co můžeme studovat na obrázcích?

- ▶ Segmentační masky (kde jsou objekty zájmu)
- ▶ Klasifikace objektů (označování)



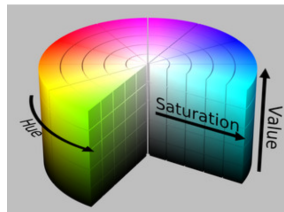
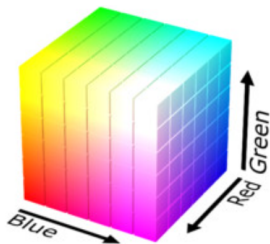
# Segmentační masky - prahování barev

## ▶ Prahování

- ▶ hodnoty pixelů RGB pro souřadnice  $\mathbf{u}$ :  $I_{\text{RGB}}(\mathbf{u})$
- ▶  $M(\mathbf{u}) = 1$ , pokud  $I_{\text{RGB}}(\mathbf{u}) = (0 \ 255 \ 0)^{\top}$  ?
- ▶  $M(\mathbf{u}) = 1$ , pokud  $\tau_l < I_{\text{RGB}}(\mathbf{u}) < \tau_u$ , pro všechny kanály
- ▶  $M(\mathbf{u}) = 1$ , pokud  $\varphi_l < I_{\text{HSV}}(\mathbf{u}) < \varphi_u$ , pro všechny kanály

## ▶ Následné zpracování

- ▶ výpočet spojených komponent
- ▶ odstranit malé nebo deformované segmenty
- ▶ přiřadit označení na základě prahových hodnot

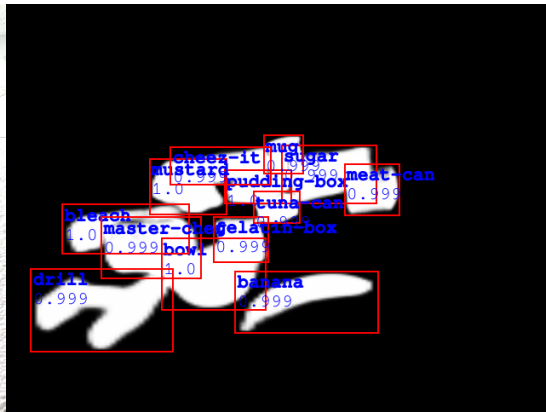


# Segmentační masky pro známé 3D objekty

- ▶ Neuronová síť (např. Mask R-CNN)
- ▶ Trénovací vstupy:
  - ▶ datová sada obrázků, masek a štítků nebo
  - ▶ datová sada známých 3D objektů (mesh)
  - ▶ kvalita závisí na trénovacích datech (augmentace)
- ▶ Inference:
  - ▶ Vstup: obrázek
  - ▶ Výstup: segmentační maska, ohraničující rámeček, označení a konfidence

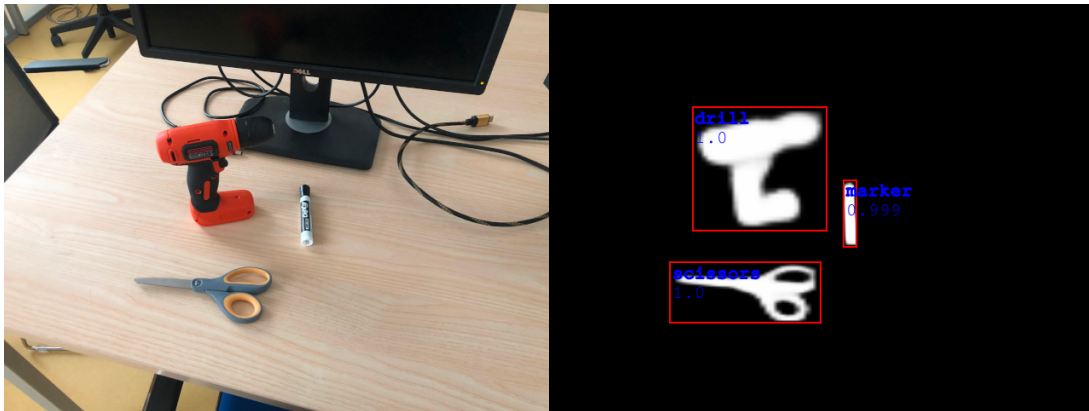


# Mask R-CNN výsledky





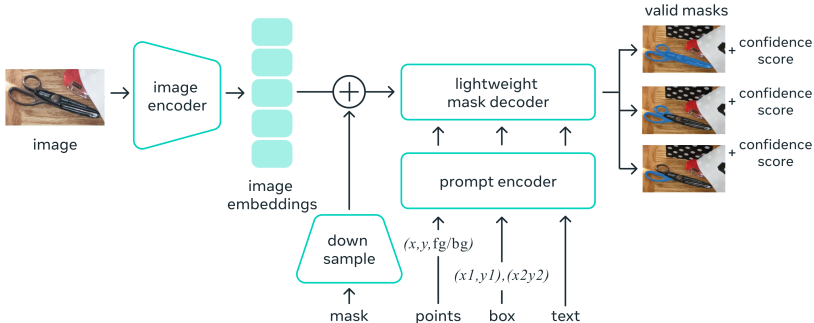
# Mask R-CNN výsledky



# Segmentační masky bez pretrénování

- ▶ Segment Anything Model (SAM)
  - ▶ segmentace libovolného objektu na libovolném snímku jediným kliknutím
  - ▶ datová sada 10 milionů obrázků, 1 miliarda masek

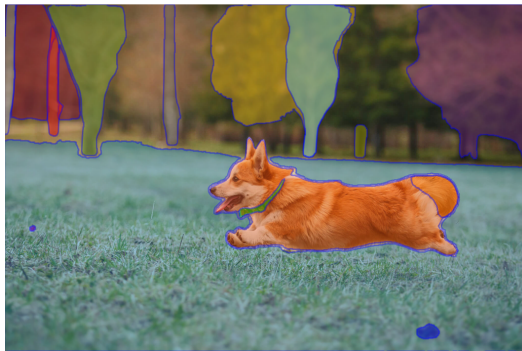
## Universal segmentation model



## SAM výsledky



## SAM výsledky



# Segmentace

- ▶ Segmentace nalezne objekty v obraze
  - ▶ segmentační maska
  - ▶ bounding box (ohraničující rámeček)
  - ▶ label (označení)
  - ▶ skóre důvěryhodnosti/konfidence
- ▶ Informace pouze v obrazovém prostoru
- ▶ Jak používat v prostoru robota?



# Externí kamera

- ▶ Předpokládáme kameru pevně připevněnou k referenčnímu rámci
  - ▶ pokud známe  $K$  a  $T_{RC}$ , jak promítnout body  $x_R$  do obrazu?
- ▶ Neznámé  $K$  a  $T_{RC}$  a planární problém
  - ▶ např. kostky se stejnou výškou na stole
  - ▶ jaká je poloha kostky na 2D stole vzhledem k 2D souřadnicím obrazu/pixelů?
  - ▶ analyzováno pomocí **homografie**

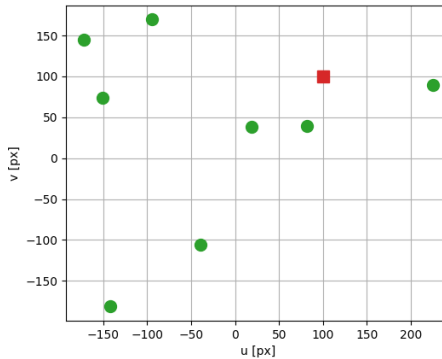
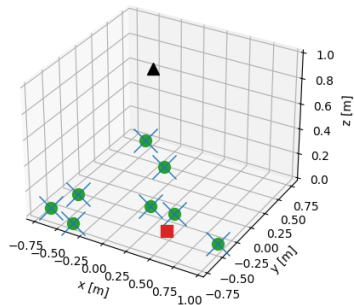


# Homografie

- ▶ Homografická matice  $H$  je matice  $3 \times 3$ , která mapuje body z jedné roviny do druhé
  - ▶ obrazová rovina na stůl
  - ▶ jedna obrazová rovina do jiné obrazové roviny (jiný pohled)
- ▶  $s \begin{pmatrix} x & y & 1 \end{pmatrix}^T = H \begin{pmatrix} u & v & 1 \end{pmatrix}^T$ 
  - ▶  $x, y$  jsou souřadnice v první rovině
  - ▶  $u, v$  jsou souřadnice ve druhé rovině
- ▶ 9 prvků, ale pouze 8 DoF, obvykle se přidává omezení  $h_{33} = 1$
- ▶ Jak najít  $H$ ?
  - ▶  $H, _ = \text{cv2.findHomography}(U, X)$
  - ▶  $U, X$  jsou  $N \times 2$  korespondenční body
  - ▶ např. ručně změříme
    - ▶ poloha středu krychle vůči rohu stolu
    - ▶ pozice středu krychle v obrázku



# Příklad homografie





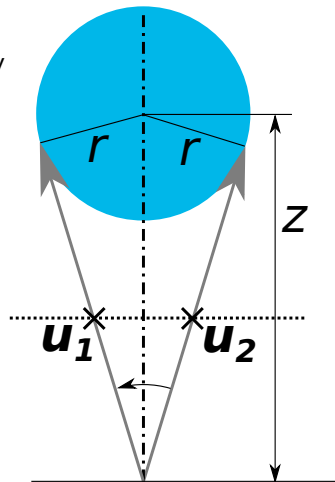
# Neplanární odhad polohy/orientace

- ▶ Homografie mapuje pouze rovinu na rovinu
- ▶ Obecnější odhad polohy objektu ve s.s **kamery**
  - ▶ získání hloubky mapováním z plochy v pixelech na hloubku pro objekty pevné velikosti
  - ▶ získání hloubky pomocí dodatečných informací o scéně, např. známá velikost/model objektu
  - ▶ RGBD kamera
  - ▶ dodatečné značky (markery)



## Využití znalostí o velikosti

- ▶ Víme, že poloměr je fixní
- ▶ Z detekovaných pixelů  $\mathbf{u}_1, \mathbf{u}_2$ , můžeme vypočítat paprsky  $\mathbf{x}_1, \mathbf{x}_2$ :  $\frac{1}{\lambda_i} \mathbf{x}_i = K^{-1} \mathbf{u}_i$
- ▶ Úhel mezi vektory:  $\cos \alpha = \frac{\frac{1}{\lambda_1 \lambda_2} \mathbf{x}_1 \cdot \mathbf{x}_2}{\frac{1}{\lambda_1 \lambda_2} \|\mathbf{x}_1\| \|\mathbf{x}_2\|}$
- ▶ Hloubka:  $z = \frac{r}{\sin(\alpha/2)}$



# Použití hloubkového senzoru

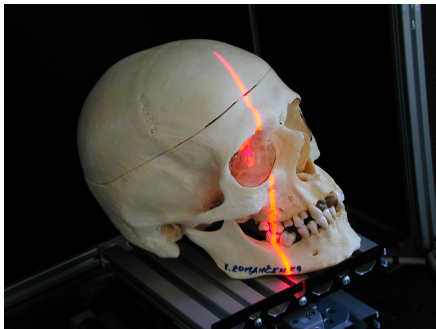
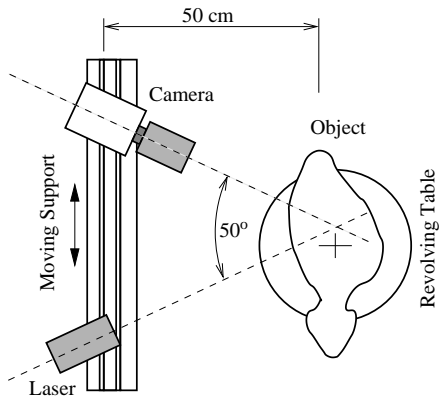
## ▶ RGBD senzory

- ▶ RGB obrázek ( $H \times W \times 3$ )
- ▶ Hloubková mapa ( $H \times W \times 1$ ), vzdálenost v metrech pro každý pixel
- ▶ Strukturované mračno bodů ( $H \times W \times 3$ ),  $(x_c \ y_c \ z_c)$  pro každý pixel



# Jak funguje hloubkový senzor

- ▶ Laser promítá vzor a kamera ho rozpozná
- ▶ Informace o hloubce se vypočítá pomocí triangulace



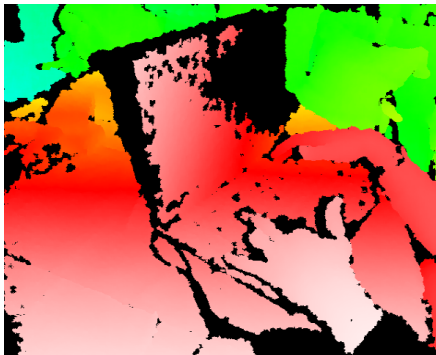
## 2D hloubkové senzory

- ▶ Na základě strukturovaného světla
- ▶ Projektuje 2D infračervené vzory
- ▶ Jeden projektor a dvě kamery (RGB + IR)



## Problémy se snímači hloubky

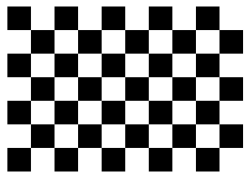
- ▶ Rekonstrukce hloubky není dokonalá (černé oblasti na snímku<sup>3</sup>)
- ▶ V jazyce python reprezentováno hodnotou NaN
- ▶ Ne každý pixel v RGB má rekonstruovanou hodnotu hloubky
- ▶ Data RGB a hloubky nejsou zarovnána (je třeba je kalibrovat)



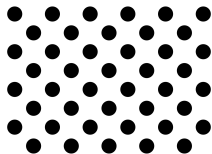
<sup>3</sup><https://commons.wikimedia.org>, User:Kolossos

## Dodatečné značky

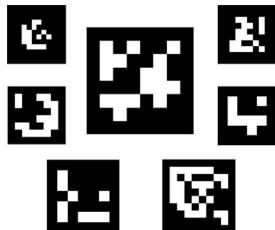
- ▶ Můžeme vypočítat polohu/orientaci obrazců<sup>4</sup>?
  - ▶ je třeba znát velikost a strukturu
  - ▶ subpixelová přesnost
  - ▶ musí být zcela viditelný
- ▶ Můžeme vypočítat polohu značek ArUco?
  - ▶ méně přesné než pravidelné obrazce
  - ▶ poskytuje identifikátor značky a pózu
  - ▶ musí být zcela viditelný



© 2008 Intel Corporation

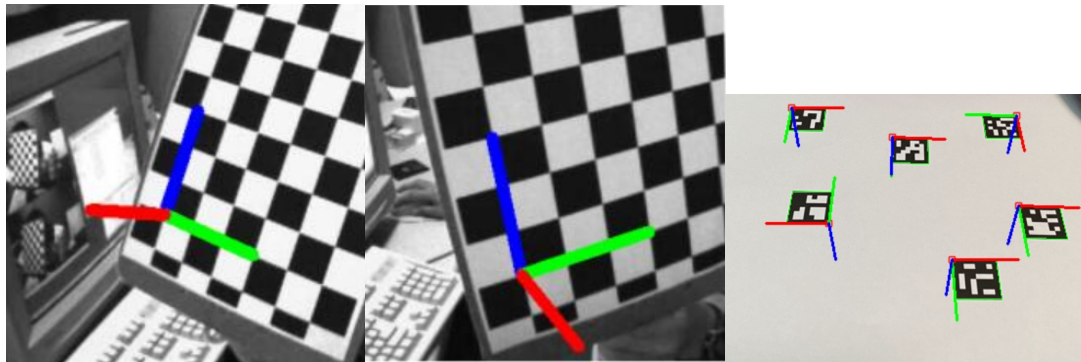


© 2008 Intel Corporation



<sup>4</sup>[docs.opencv.org](http://docs.opencv.org)

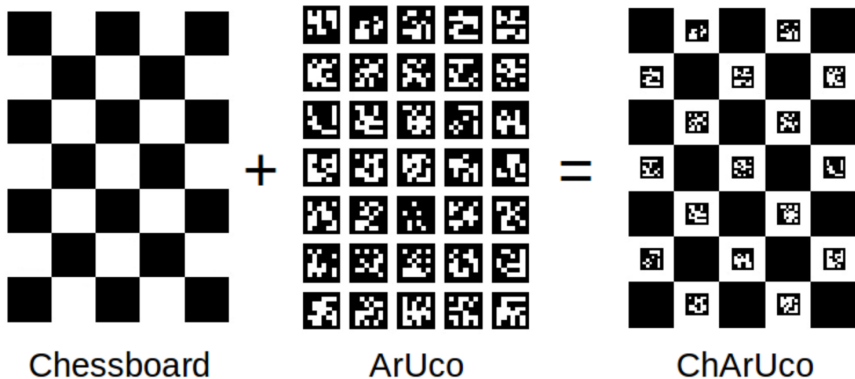
## Příklad umístění značek





## ChArUco deska pro kalibraci

- ▶ Kombinuje přesnost pravidelného vzoru s detekcí ArUco
- ▶ Detekce dílčích částí / částečná viditelnost



# Odhad matice kamery pomocí kalibračních desek

- ▶ Můžeme odhadnout matici kamery z korespondencí v obrazovém a prostorovém prostoru
  - ▶ nasbíráme snímky tabule z různých pohledů
  - ▶ detekujeme desky
  - ▶ vypočítáme korespondence mezi obrazovými body a body s.s. tabule
  - ▶ `_, K, dist_coeffs, rvecs, tvecs = cv2.calibrateCamera(obj_points, img_points, img_shape)`
- ▶ Kromě toho získáme
  - ▶ koeficienty zkreslení, které kompenzují vady objektivu  
`Knew, roi = cv.getOptimalNewCameraMatrix(K, dist_coeffs, img_shape, 1, img_shape)`  
`img_undistorted = cv.undistort(img, K, dist_coeffs, None, Knew)`
  - ▶  $SE(3)$  pozice desek v s.s. kamery



# Odhad pozice z RGB(D)

- ▶ Metody odhadu polohy
  - ▶ používají předchozí znalosti o úloze, např. objekty s pevnou výškou v rovině
  - ▶ používají předchozí znalosti o objektech (velikost)
  - ▶ používají hloubkový senzor
  - ▶ používají značky ArUco
- ▶ Kde je robot?
  - ▶ homografie odhaduje polohy objektů vzhledem k s.s. rovině
  - ▶ ostatní metody odhadují polohy v s.s. kamery
  - ▶ potřebujeme odhadnout/kalibrovat  $T_{RC}$



## Kalibrace RukaOkó (HandEye)

- ▶ Kamera může být namontována na
  - ▶ základně robota (kalibrace oko-ruka, eye-to-hand)
  - ▶ chapadle (kalibrace oko-v-ruce, eye-in-hand)
- ▶ Řešíme  $A^i X = Y B^i$ 
  - ▶ měření:  $A^i, B^i \in SE(3)$
  - ▶ odhadnuté parametry:  $X, Y \in SE(3)$
- ▶  $X, Y = \text{calibrateRobotWorldHandEye}(A, B)$
- ▶ Kalibrace oko-ruka, eye-to-hand
  - ▶  $A^i = T_{RG}^i$
  - ▶  $B^i = T_{CT}^i$
  - ▶  $X = T_{GT}$
  - ▶  $Y = T_{RC}$
- ▶ Kalibrace oko-v-ruce, eye-in-hand
  - ▶  $A^i = T_{CT}^i$
  - ▶  $B^i = T_{GR}^i$
  - ▶  $X = T_{TR}$
  - ▶  $Y = T_{CG}$



# Shrnutí

- ▶ Repräsentace obrázku
- ▶ Promítání do/z obrazu
- ▶ Segmentace v obrazovém prostoru
- ▶ Homografie
- ▶ Odhad polohy z obrazu
- ▶ Kalibrace kamery



# Cvičení

- ▶ Tento týden žádný nový domácí úkol
- ▶ Odhad homografie na příkladu v Pythonu/OpenCV
- ▶ Kalibrace oko-ruka na příkladu v Pythonu/OpenCV

