

Robotics: Introduction to perception

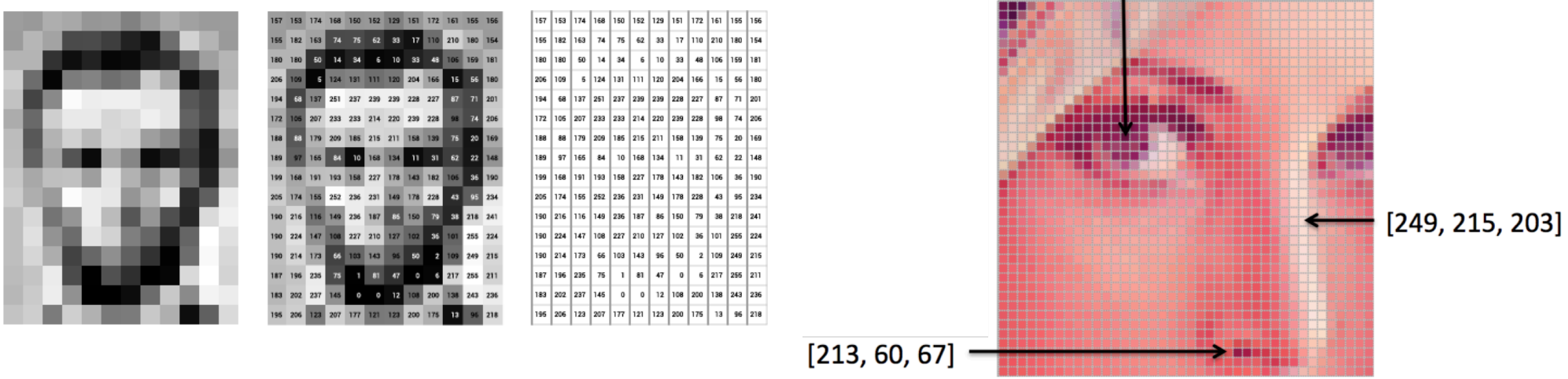
Vladimír Petrík

vladimir.petrik@cvut.cz

12 October 2026

What is an image?

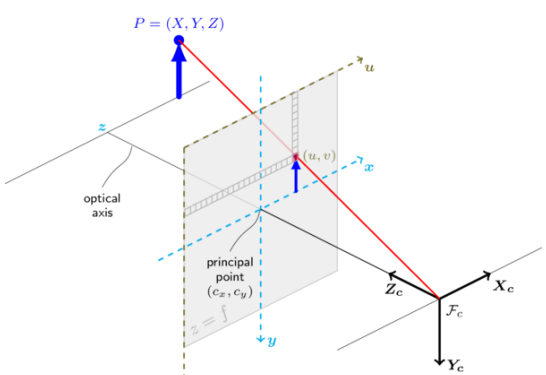
- A camera connected to a computer produces images
- An image is an array of numbers

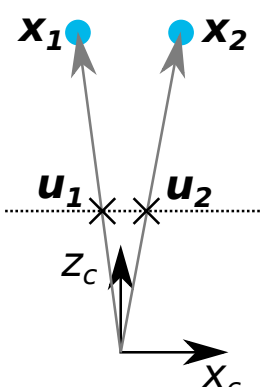


Images are from: https://ai.stanford.edu/~syyeung/cvweb/tutorial1.html

How is the image formed?

- Perspective camera
  - pinhole camera model (docs.opencv.org)
  - projects a spatial point $\boldsymbol{x}_c$ into an image point $\boldsymbol{u} = (u, v)^\top$ by intersecting
    - the image plane and
    - the line connecting $\boldsymbol{x}_c$ with the projection center
  - all points on a ray project to the same pixel





Projection of the pinhole camera

- $\boldsymbol{u}_H = K\boldsymbol{x}_c$
  - $\boldsymbol{u}_H$ is the pixel in homogeneous coordinates
  - if $\boldsymbol{u}_H = (u_H, v_H, w_H)^\top$, then the pixel coordinates are $(u_H/w_H, v_H/w_H)^\top$
  - alternatively, we can represent it as: $\lambda(u, v, 1)^\top = K\boldsymbol{x}_c$
- K is the camera matrix

$$K = \begin{pmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{pmatrix}$$

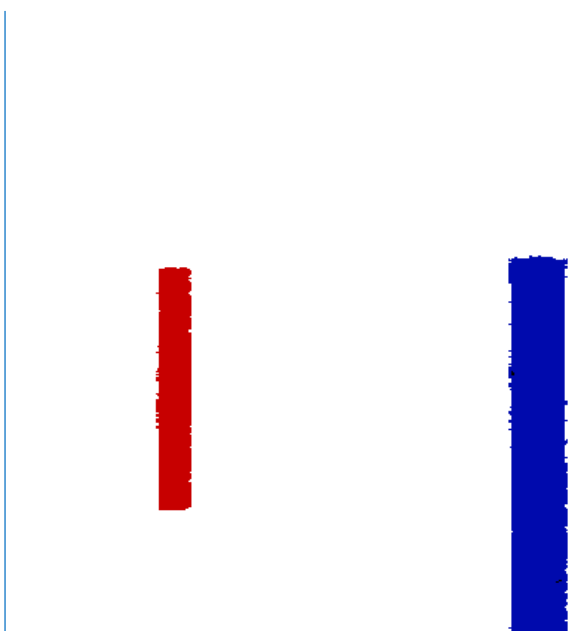
  - what does λ represent?
    - λ is a non-zero real number
    - if you know the value of λ, you can compute the Cartesian coordinates $\boldsymbol{x} = \lambda K^{-1}\boldsymbol{u}$
    - otherwise, only the ray is computable
  - how to find K from points?

What can we study on images?

- Segmentation masks (where are the objects of interest)
- Object classification (labeling)



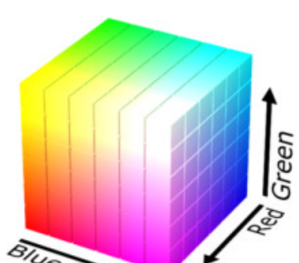


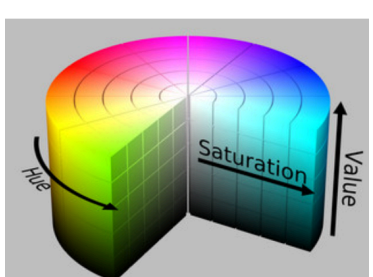


Segmentation masks – color thresholding

- Thresholding
  - RGB pixel values at coordinates $\boldsymbol{u}$: $I_{\mathrm{RGB}}(\boldsymbol{u})$
  - $M(\boldsymbol{u}) = 1$, if $I_{\mathrm{RGB}}(\boldsymbol{u}) = (0, 255, 0)^\top$?
  - $M(\boldsymbol{u}) = 1$, if $\boldsymbol{\tau}_l < I_{\mathrm{RGB}}(\boldsymbol{u}) < \boldsymbol{\tau}_u$, for all channels
  - $M(\boldsymbol{u}) = 1$, if $\boldsymbol{\varphi}_l < I_{\mathrm{HSV}}(\boldsymbol{u}) < \boldsymbol{\varphi}_u$, for all channels
- Post-processing
  - compute connected components
  - remove small or deformed segments
  - assign a label based on the thresholds





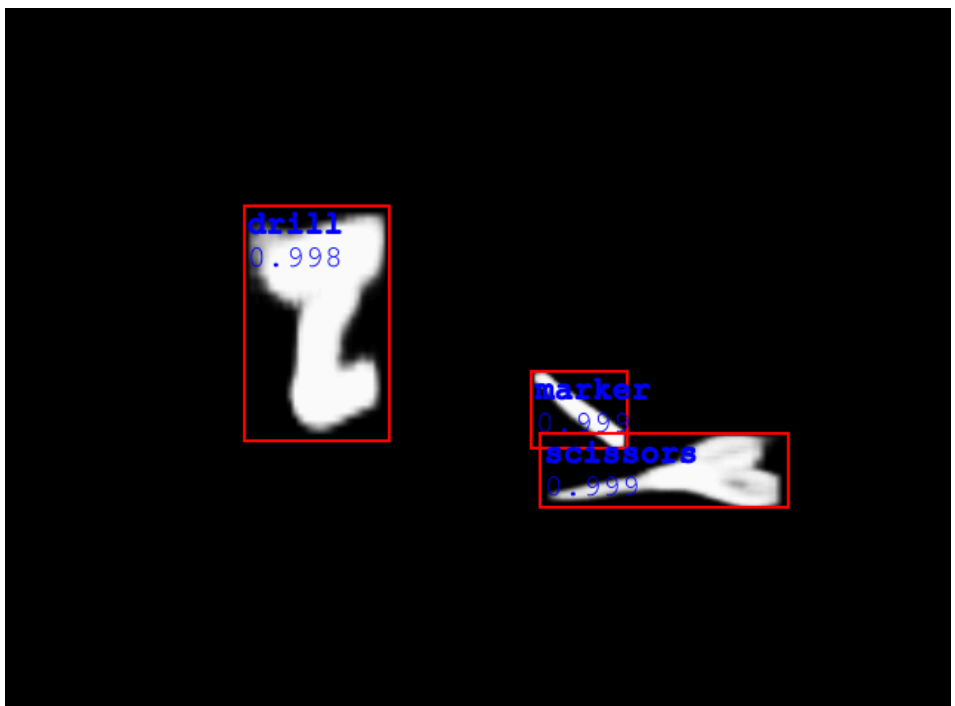


Segmentation masks for known 3D objects

- Neural network (e.g. Mask R-CNN)
- Training inputs:
  - dataset of images, masks and labels, or
  - dataset of known 3D objects (meshes)
  - quality depends on the training data (augmentations)
- Inference:
  - Input: image
  - Output: segmentation mask, bounding box, label, and confidence

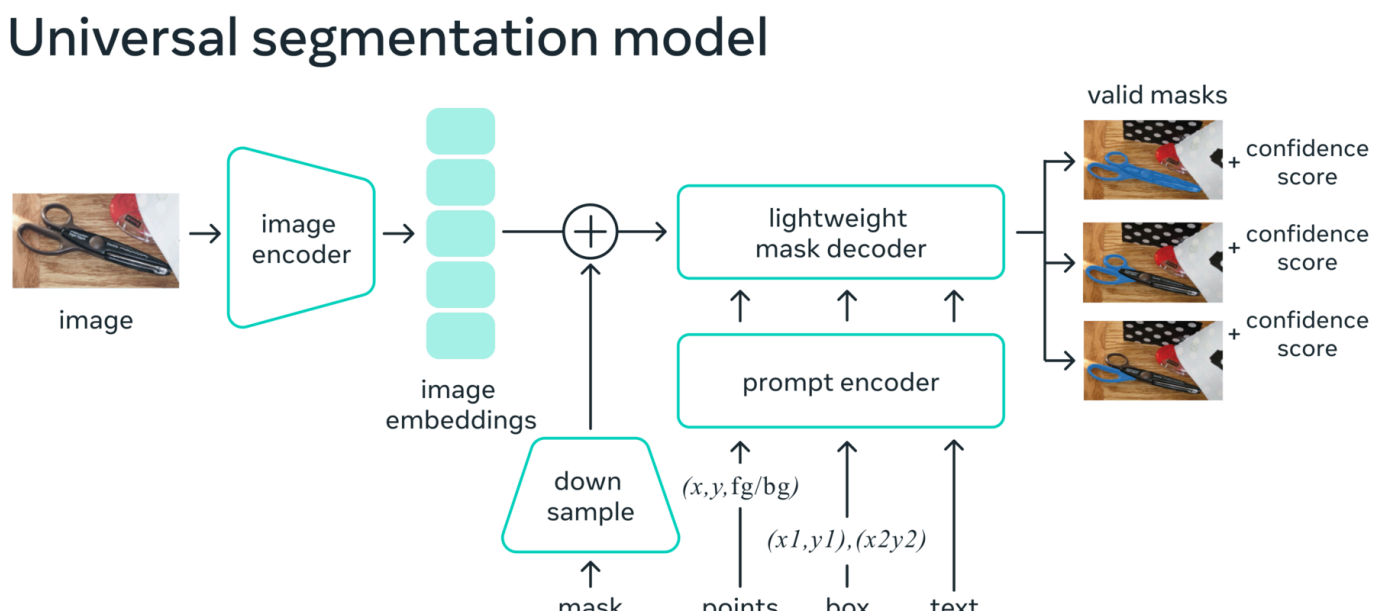
Mask R-CNN results





Segmentation masks without re-training

- Segment Anything Model (SAM)
  - segment any object, in any image, with a single click
  - dataset of 10M images, 1B masks



SAM results



SAMv2



Segmentation

- Segmentation finds objects in the image
  - segmentation mask
  - bounding box
  - label
  - confidence score
- Information only in image space
- How to use it in robot space?

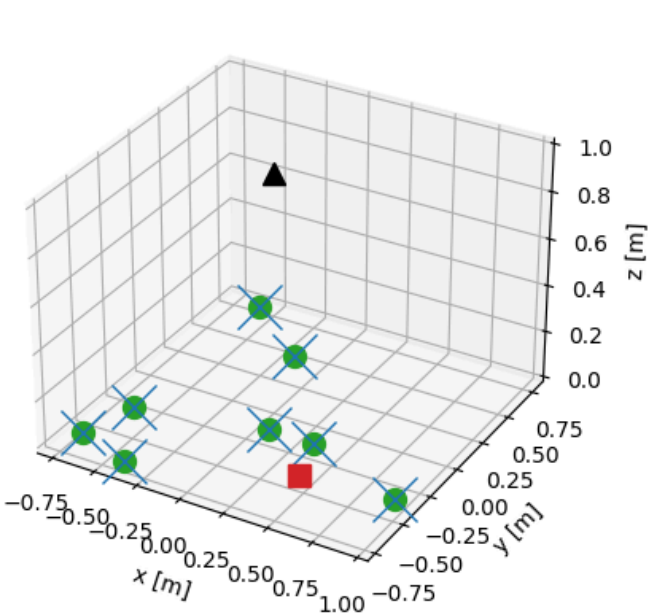
External camera

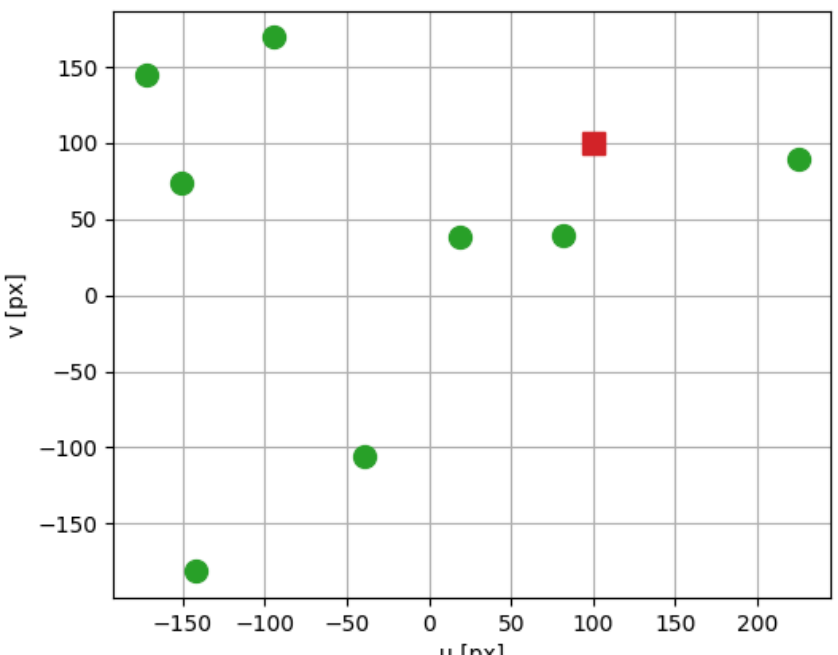
- Assume the camera is mounted rigidly to the reference frame
  - if we know K and T_{RC}, how to project points x_R to the image?
- Unknown K and T_{RC} and a planar problem
  - e.g. cubes of the same height on a table
  - what is the position of a cube on the 2D table w.r.t. the 2D image/pixel coordinates?
  - analyzed by **homography**

Homography

- The homography matrix H is a 3×3 matrix that maps points from one plane to another
  - image plane to table
  - one image plane to another image plane (different view)
- $s(x, y, 1)^\top = H(u, v, 1)^\top$
  - x, y are coordinates in the first plane
  - u, v are coordinates in the second plane
- 9 elements but only 8 DoF, usually the constraint $h_{33} = 1$ is added
- How to find H?
  - `H, _ = cv2.findHomography(U, X)`
  - U, X are $N \times 2$ correspondence points
  - e.g. measure manually
    - position of the cube center w.r.t. the table corner
    - position of the cube center in the image

Homography example





Non-planar pose estimation

- Homography maps only a plane to a plane
- More general object pose estimation in the **camera** frame
  - get depth by mapping from the area in pixels to depth for fixed-size objects
  - get depth from additional scene information, e.g. known size/model of the objects
  - RGBD camera
  - additional markers

Using prior knowledge about size

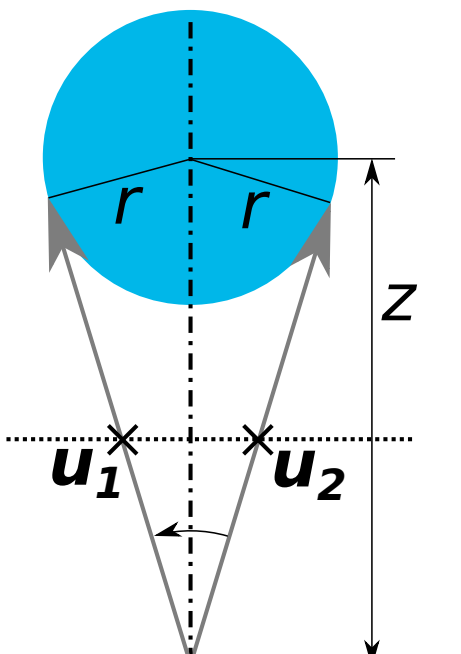
- We know the radius r is fixed
- From the detected pixels u_1, u_2 we can compute the rays x_1, x_2:

$$\frac{1}{\lambda_i} x_i = K^{-1} u_i$$

- Angle between the vectors (independent of λ_1, λ_2):

$$\cos \alpha = \frac{x_1 \cdot x_2}{\|x_1\| \|x_2\|}$$

- Depth: $z = \frac{r}{\sin(\alpha/2)}$



Using a depth sensor

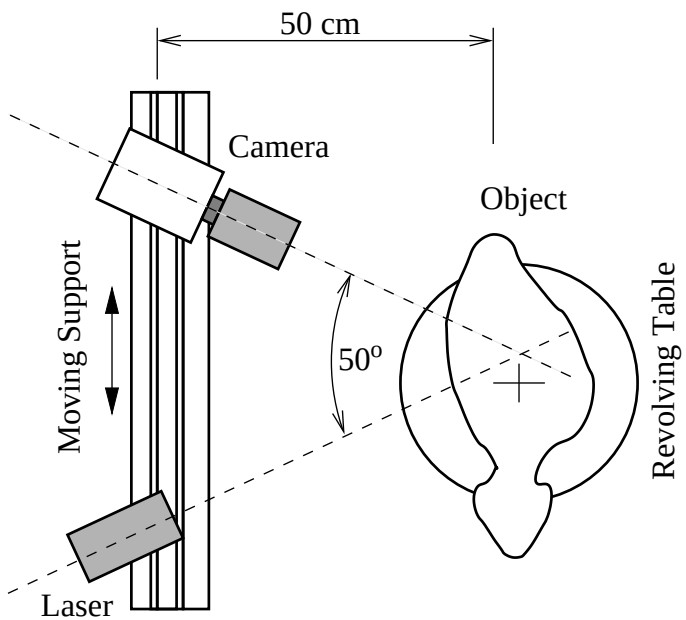
- RGB**D** sensors
  - RGB image ($H \times W \times 3$)
  - Depth map ($H \times W \times 1$), distance in meters for each pixel
  - Structured point cloud ($H \times W \times 3$), (x_c, y_c, z_c) for each pixel

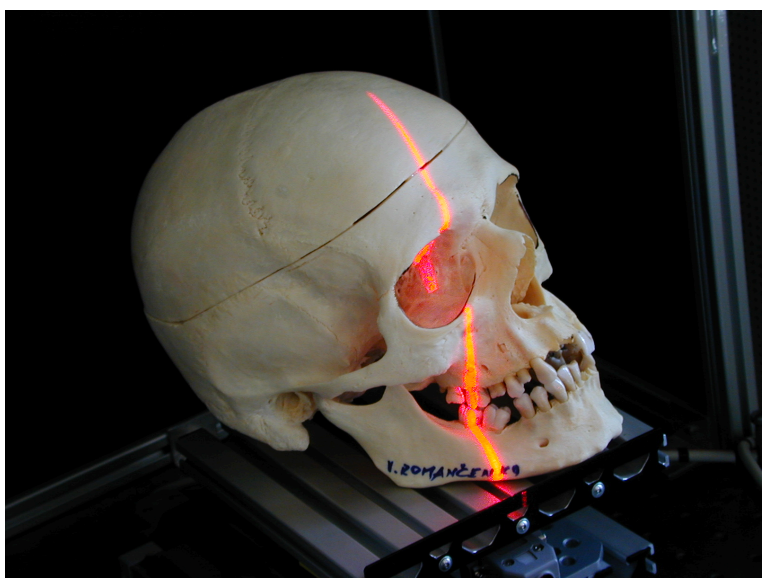




How a depth sensor works

- A laser projects a pattern and the camera recognizes it
- The depth information is computed using triangulation





2D depth sensors

- Based on structured light
- Project 2D infrared patterns
- One projector and two cameras (RGB + IR)

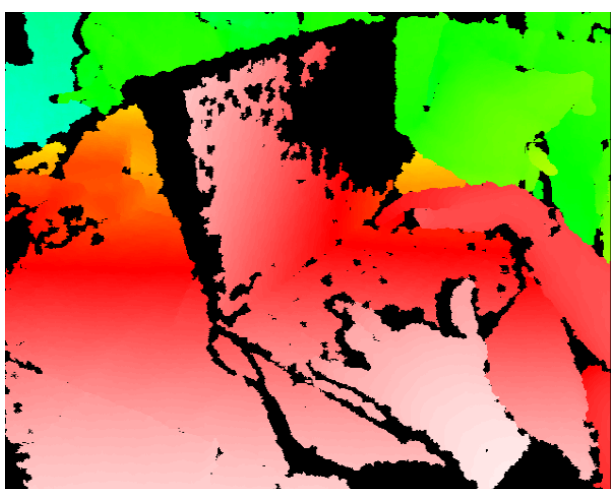




Issues with depth sensors

- Depth reconstruction is not perfect (black areas in the image)
- In Python represented by NaN
- Not every pixel in RGB has a reconstructed depth value
- RGB and depth data are not aligned (you need to calibrate them)

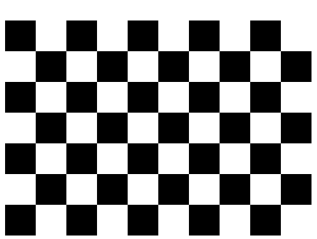


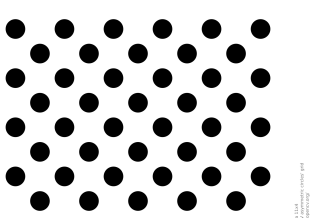


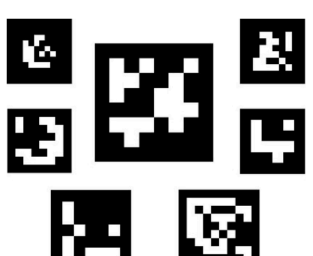
commons.wikimedia.org, User:Kolossos

Additional markers

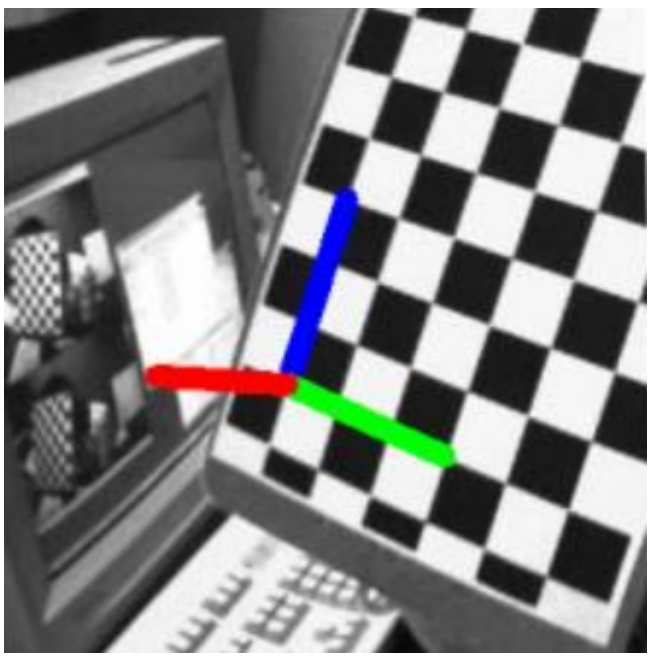
- Can we compute the pose of patterns? (docs.opencv.org)
  - the size and structure need to be known
  - subpixel accuracy
  - it has to be completely visible
- Can we compute the pose of ArUco markers?
  - less accurate than regular patterns
  - provide the marker id and the pose
  - it has to be completely visible

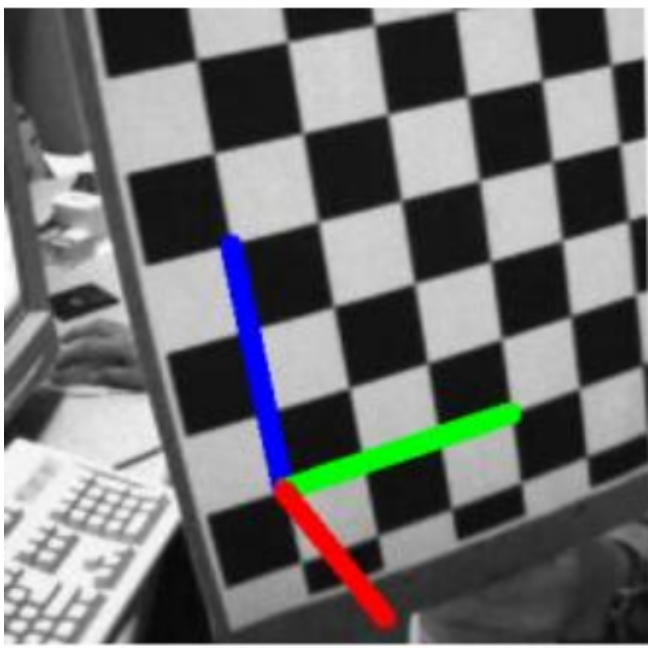


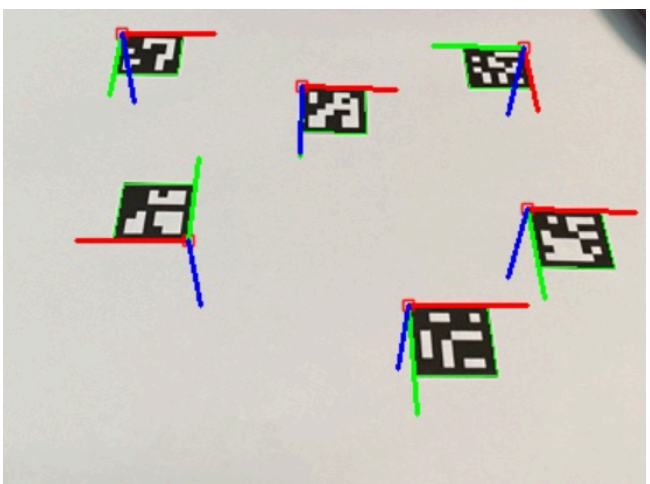




Marker pose example





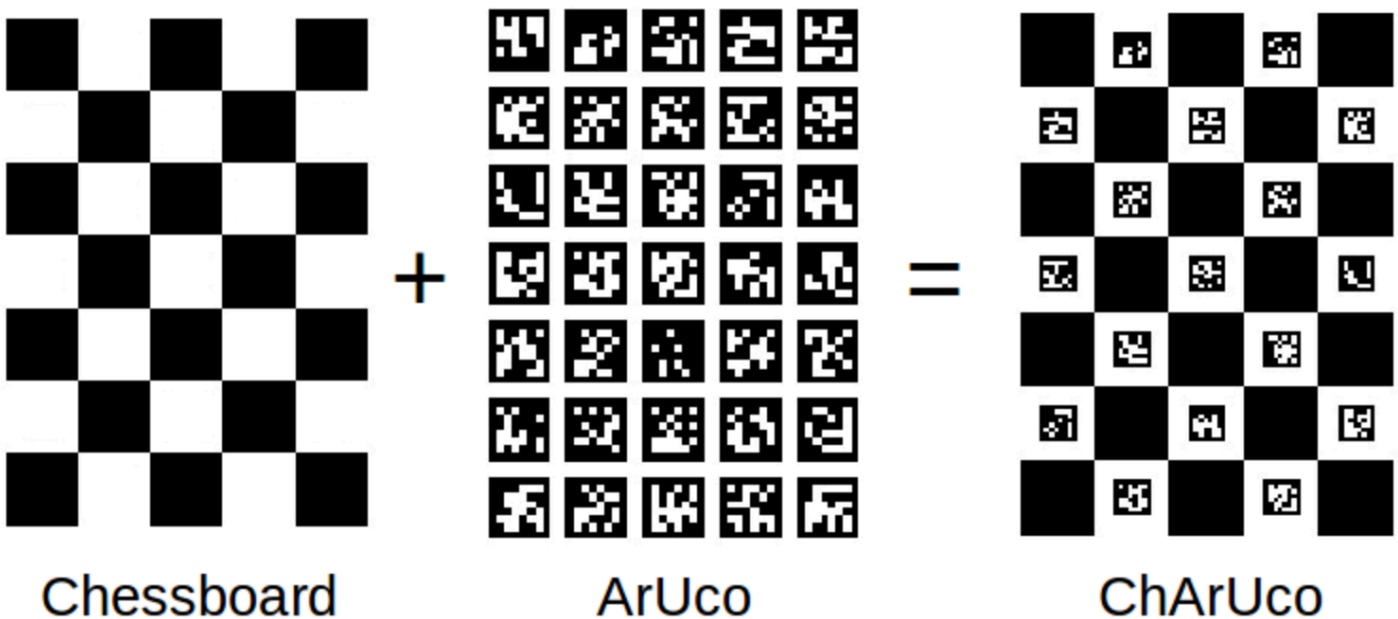


Markers example from the real world



ChArUco board for calibration

- Combines the accuracy of a pattern with the detection of ArUco
- Detection under partial visibility



Charuco definition

Camera matrix estimation with boards

- We can estimate the camera matrix from correspondences between image space and 3D space
  - collect images of the board from different views
  - detect the boards
  - compute correspondences between image points and board-frame points
  - `_, K, dist_coeffs, rvecs, tvecs = cv2.calibrateCamera(obj_points, img_points, img_shape)`
- In addition we get
  - distortion coefficients that compensate the defects of the lens
  - ```
    Knew, roi = cv2.getOptimalNewCameraMatrix(K, dist_coeffs, img_shape, 1, img_shape)
    img_undistorted = cv2.undistort(img, K, dist_coeffs, None, Knew)
    ```
  - $SE(3)$ poses of the boards in the camera frame
    ```

Pose estimation from RGB(D)

- Pose estimation methods
  - use prior knowledge about the task, e.g. fixed-height objects on a plane
  - use prior knowledge about the objects (size)
  - use a depth sensor
  - use ArUco markers
- Where is the robot?
  - homography estimates the poses of objects w.r.t. the plane frame
  - other methods estimate poses in the camera frame
  - we need to estimate/calibrate T_{RC}

Summary

- Image representation
- Projection to/from the image
- Segmentation in image space
- Homography
- Pose estimation from the image
- Camera calibration

Laboratory

- Camera–robot transformation, homography (hw03, due 25. 10.)
- Final project assignment: see the course page

What is next?

- Next week, we will start with a test
  - questions from the first three lectures
  - $SE(2)$ and $SE(3)$ transformations and their properties
  - forward kinematics and DoF