



Robotics: Calibration

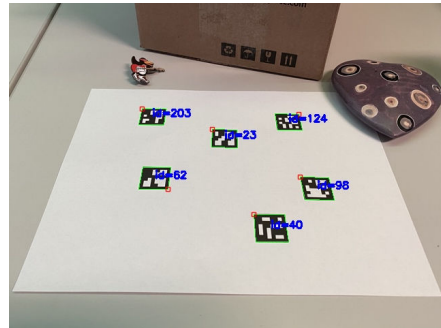
Vladimír Petřík

vladimir.petrík@cvut.cz

13.10.2025

What we know¹

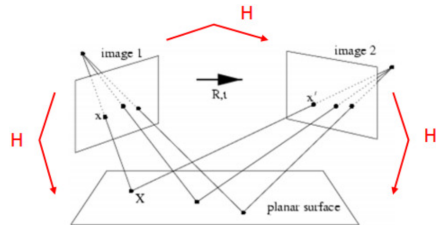
- ▶ How to detect objects in image
 - ▶ Checkerboard
 - ▶ Aruco markers
- ▶ How to estimate Homography
 - ▶ Maps points between planes
 - ▶ Estimated from correspondences
 - ▶ Robust - no depth estimation
- ▶ Camera intrinsics calibration
 - ▶ Use checkerboard images
 - ▶ Estimate camera matrix K
- ▶ Pose estimation
 - ▶ Estimates camera-object pose
 - ▶ Uses PnP algorithm
 - ▶ Requires K and correspondences



¹Images from docs.opencv.org

What we know¹

- ▶ How to detect objects in image
 - ▶ Checkerboard
 - ▶ Aruco markers
- ▶ How to estimate Homography
 - ▶ Maps points between planes
 - ▶ Estimated from correspondences
 - ▶ Robust - no depth estimation
- ▶ Camera intrinsics calibration
 - ▶ Use checkerboard images
 - ▶ Estimate camera matrix K
- ▶ Pose estimation
 - ▶ Estimates camera-object pose
 - ▶ Uses PnP algorithm
 - ▶ Requires K and correspondences

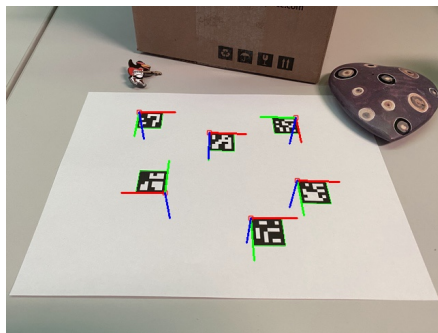
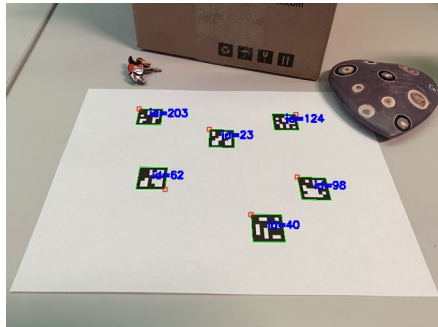


¹Images from docs.opencv.org

What we know¹

- ▶ How to detect objects in image
 - ▶ Checkerboard
 - ▶ Aruco markers
- ▶ How to estimate Homography
 - ▶ Maps points between planes
 - ▶ Estimated from correspondences
 - ▶ Robust - no depth estimation
- ▶ Camera intrinsics calibration
 - ▶ Use checkerboard images
 - ▶ Estimate camera matrix K
- ▶ Pose estimation
 - ▶ Estimates camera-object pose
 - ▶ Uses PnP algorithm
 - ▶ Requires K and correspondences

¹Images from docs.opencv.org



Where is robot?

- ▶ Consider homography - how we can get 3D points in robot frame?
 1. Manually measure Aruco positions with respect to the base frame
 - ▶ Place Aruco markers in the plane of interest
 - ▶ Might be hard to measure accurately
 2. Manually position robot to calibration target
 - ▶ Compose 3D points of the target with FK
 3. Use an Aruco marker grasped by the robot
 - ▶ Use FK to compute position of the marker in the robot frame
 - ▶ We need to move in the plane of interest
 - ▶ We will not have precise estimation of marker w.r.t. hand
- ▶ Limited to plane-to-plane mapping



Calibrating camera pose

- ▶ Hand-eye calibration
- ▶ Solve $A^i X = Y B^i$
 - ▶ Measurements: $A^i, B^i \in SE(3)$
 - ▶ Estimated parameters: $X, Y \in SE(3)$
- ▶ Camera can be mounted w.r.t.
 - ▶ Robot base frame (eye-to-hand calibration)
 - ▶ Gripper frame (eye-in-hand calibration)
- ▶ Eye-to-hand calibration
 - ▶ $A^i = T_{RG}^i$
 - ▶ $B^i = T_{CT}^i$
 - ▶ $X = T_{GT}$
 - ▶ $Y = T_{RC}$
- ▶ Eye-in-hand calibration
 - ▶ $A^i = T_{CT}^i$
 - ▶ $B^i = T_{GR}^i$
 - ▶ $X = T_{TR}$
 - ▶ $Y = T_{CG}$



Code for $AX = YB$ calibration

```
def solve_AX_YB(a: list[SE3], b: list[SE3]) -> tuple[SE3, SE3]:  
    """Solve  $A^iX=YB^i$ , return X, Y"""  
  
    rvec_a = [T.rotation.log() for T in a]  
    tvec_a = [T.translation for T in a]  
    rvec_b = [T.rotation.log() for T in b]  
    tvec_b = [T.translation for T in b]  
  
    Rx, tx, Ry, ty = cv2.calibrateRobotWorldHandEye(rvec_a, tvec_a, rvec_b, tvec_b)  
    return SE3(tx[:, 0], SO3(Rx)), SE3(ty[:, 0], SO3(Ry))
```



Influence of Aruco marker size

- ▶ Depth estimation of small Aruco markers is hard
 - ▶ Large errors results in imprecise calibration
 - ▶ Large markers limit the motion
- ▶ Solution: minimization of reprojection errors
 - ▶ Original measurements are in pixels
 - ▶ We can perform fine optimization in pixel space
 - ▶ Reprojection error:
 - ▶ $X^*, Y^* = \arg \min_{X, Y} (\sum_i \mathbf{u}_i - \pi(\mathbf{x}_i(X, Y)))$
 - ▶ $\mathbf{u}_i$ - detected 2D point, $\mathbf{x}_i$ - 3D point in camera frame, π - projection
 - ▶ $\mathbf{x}_i$ is computed by FK and X and Y poses
 - ▶ Initial guess is from Hand-eye calibration



Influence of robot kinematics

- ▶ Robot kinematics is not perfect
 1. Different configuration of the robot has different errors
 2. Often errors in joint offsets
- ▶ Solution #1: restricts to a subset of configurations
 - ▶ When solving IK, use only similar configurations
 - ▶ For the calibration but also for the robot operation!
- ▶ Solution #2: calibrate joint offset
 - ▶ As in the reprojection error minimization
 - ▶ $X^*, Y^*, \theta_{\text{offset}}^* = \arg \min_{X, Y, \theta_{\text{offset}}^*} (\sum_i u_i - \pi(x_i(X, Y, \theta_{\text{offset}}^*)))$
 - ▶ FK is adjusted to include joint offsets
- ▶ Calibration is not a one-time task
 - ▶ Calibrate robot before each important task
 - ▶ Calibrate robot after any significant change



Conclusion

- ▶ Calibration depends on the task
 - ▶ Homography
 - ▶ Camera intrinsics
 - ▶ Camera pose
 - ▶ Robot kinematics
- ▶ Calibration is not a one-time task
 - ▶ Good to automate it



Laboratory

- ▶ Laboratory at KN
- ▶ Hoop detection and homography

