Robotics: Calibration

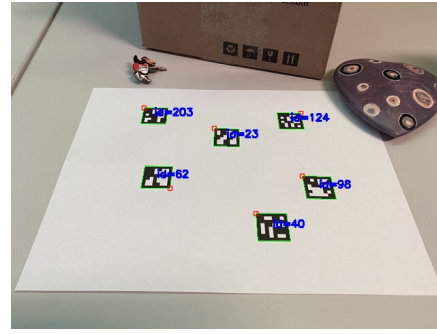
Vladimír Petrík

vladimir.petrik@cvut.cz

04.11.2024

What we know[1]

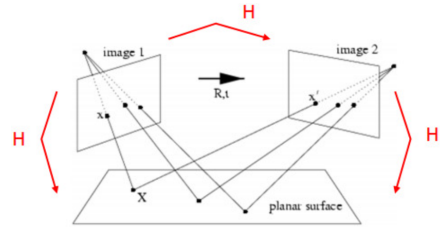
- How to detect objects in image
  - Checkerboard
  - Aruco markers



[1] Images from docs.opencv.org

What we know[1]

- How to detect objects in image
  - Checkerboard
  - Aruco markers
- How to estimate Homography
  - Maps points between planes
  - Estimated from correspondences
  - Robust - no depth estimation



[1]Images from docs.opencv.org

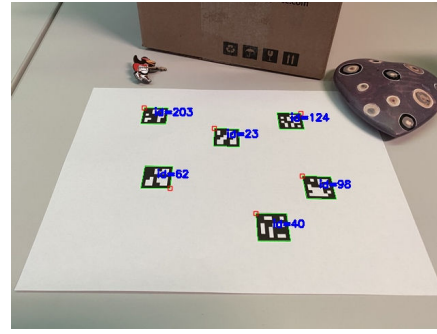
What we know[1]

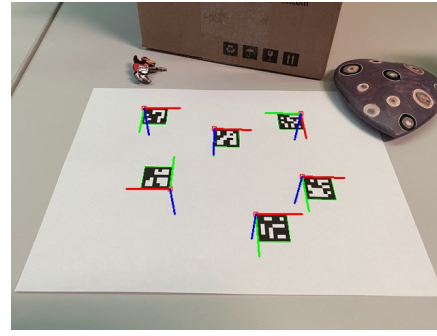
- How to detect objects in image
  - Checkerboard
  - Aruco markers
- How to estimate Homography
  - Maps points between planes
  - Estimated from correspondences
  - Robust - no depth estimation
- Camera intrinsics calibration
  - Use checkerboard images
  - Estimate camera matrix K

[1] Images from docs.opencv.org

What we know[1]

- 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





[1] Images from docs.opencv.org

Where is robot?

- Consider homogrpahy - how we can get 3D points in robot frame?

Where is robot?

- Consider homogrpahy - 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 accuratelly

Where is robot?

- Consider homogrpahy - 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 accuratelly
  2. Manually position robot to calibration target
     - Compose 3D points of the target with FK

Where is robot?

- Consider homogrpahy - 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 accuratelly
  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: $X, Y \in SE(3)$

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)

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
  - $A^i = T^i_{\mathsf{RG}}$
  - $B^i = T^i_{\mathsf{CT}}$
  - $X = T_{\mathsf{GT}}$
  - $Y = T_{\mathsf{RC}}$

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
  - $A^i = T^i_{\mathsf{RG}}$
  - $B^i = T^i_{\mathsf{CT}}$
  - $X = T_{\mathsf{GT}}$
  - $Y = T_{\mathsf{RC}}$
- Eye-in-hand calibration
  - $A^i = T^i_{\mathsf{CT}}$
  - $B^i = T^i_{\mathsf{GR}}$
  - $X = T_{\mathsf{TR}}$
  - $Y = T_{\mathsf{CG}}$

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

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:
    - $X^*, Y^* = \arg\min_{X,Y}(\sum_i \boldsymbol{u}_i - \pi(\boldsymbol{x}_i(X, Y)))$
    - $\boldsymbol{u}_i$ - detected 2D point, $\boldsymbol{x}_i$ - 3D point in camera frame, π - projection
    - $\boldsymbol{x}_i$ 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

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!

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: callibrate joint offset
  - As in the reprojection error minimization
  - $X^*, Y^*, \boldsymbol{\theta}^*_{\text{offset}} = \arg\min_{X,Y,\boldsymbol{\theta}^*_{\text{offset}}} (\sum_i \boldsymbol{u}_i - \pi(\boldsymbol{x}_i(X, Y, \boldsymbol{\theta}^*_{\text{offset}})))$
  - FK is adjusted to include joint offsets

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: callibrate joint offset
  - As in the reprojection error minimization
  - $X^*, Y^*, \boldsymbol{\theta}^*_{\text{offset}} = \arg\min_{X,Y,\boldsymbol{\theta}^*_{\text{offset}}} (\sum_i \boldsymbol{u}_i - \pi(\boldsymbol{x}_i(X, Y, \boldsymbol{\theta}^*_{\text{offset}})))$
  - 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

- Laboratories this week are **mandatory**
- Safety instructions and robot control tutorial
- Room JP-B:633 (CIIRC) in Dejvice