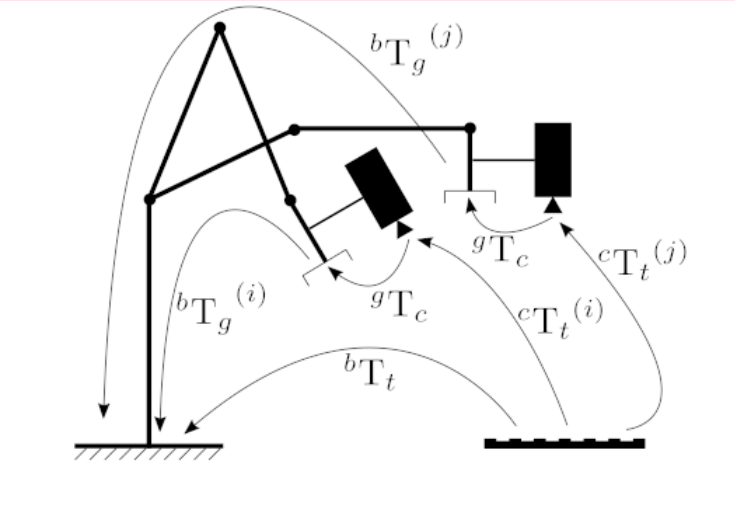
2D 비전 데이터를 기반으로 대상물 위치 검출

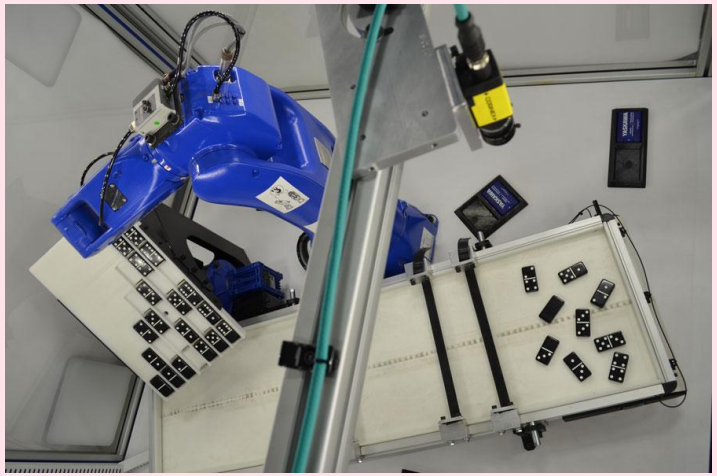
Hand-eye Calibration

Camera Calibration → Camera Pose From Aruco Marker → Robot Pose → Generate Calibration Matrix



Vision Guided Pick and place

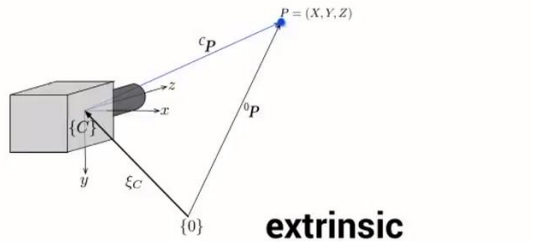
Object Detection → Camera Pose → Robot Pose w/ Calib. Mat → Pick



Camera Calibration and Intrinsic Parameters

Complete camera model

Corke, P. | Reproduced with permission from Springer Science, & Business Media



$$\begin{pmatrix}\tilde{x}\\ \tilde{y}\\ \tilde{z}\end{pmatrix} = \begin{pmatrix}f & 0 & 0 & 0\\ 0 & f & 0 & 0\\ 0 & 0 & 1 & 0\end{pmatrix}\begin{pmatrix}X\\ Y\\ Z\\ 1\end{pmatrix}$$

extrinsic parameters

$$\begin{pmatrix}\tilde{u}\\ \tilde{v}\\ \tilde{w}\end{pmatrix} = \underbrace{\begin{pmatrix}\frac{1}{\rho_u} & 0 & u_0\\ 0 & \frac{1}{\rho_v} & v_0\\ 0 & 0 & 1\end{pmatrix}\begin{pmatrix}f & 0 & 0 & 0\\ 0 & f & 0 & 0\\ 0 & 0 & 1 & 0\end{pmatrix}}_{\mathbf{K}}\underbrace{\begin{pmatrix}\mathbf{R} & t\\ \mathbf{0}_{1\times 3} & 1\end{pmatrix}^{-1}}_{\mathbf{C}}\begin{pmatrix}X\\ Y\\ Z\\ 1\end{pmatrix}$$

intrinsic parameters

camera matrix

- findChessboardCorners

```
ret, corners = cv2.findChessboardCorners(gray, CHECKERBOARD,
cv2.CALIB_CB_ADAPTIVE_THRESH + cv2.CALIB_CB_FAST_CHECK +
cv2.CALIB_CB_NORMALIZE_IMAGE)
```

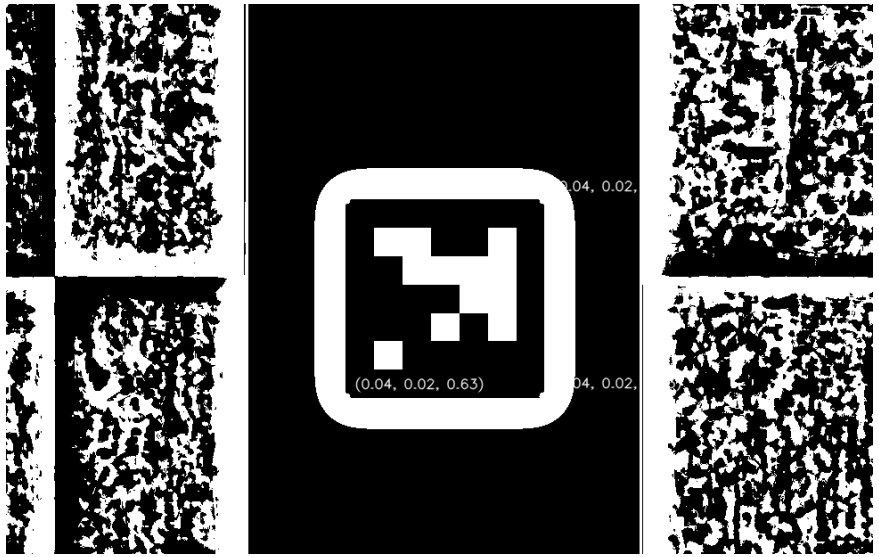
- calibrateCamera

```
ret, mtx, dist, rvecs, tvecs = cv2.calibrateCamera(objpoints,
imgpoints, gray.shape[::-1], None, None)
```

$$A = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}$$

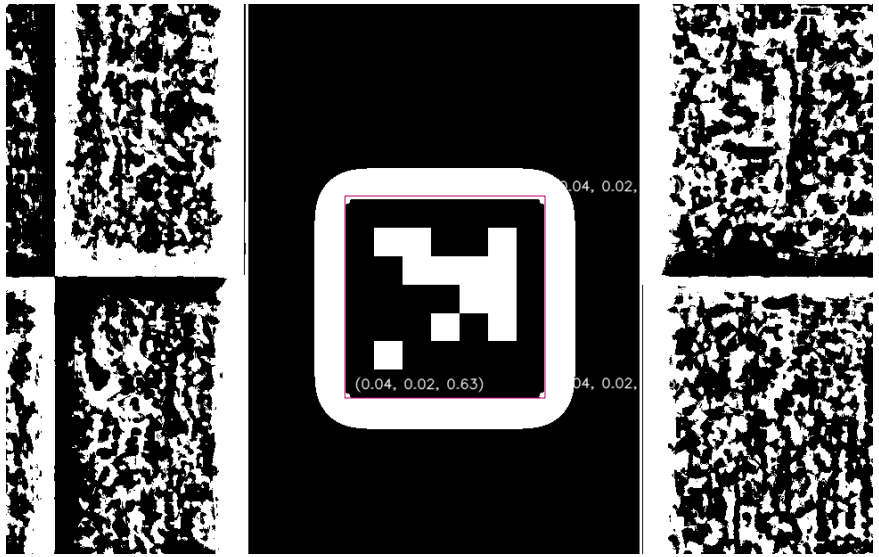
distortion coefficients $(k_1, k_2, p_1, p_2[, k_3[, k_4, k_5, k_6[, s_1, s_2, s_3, s_4[, \tau_x, \tau_y]]]])$ of 4, 5, 8, 12 or 14 elements.

Camera Pose from Aruco Marker




```
gray_image = cv2.cvtColor(image_origin, cv2.COLOR_BGR2GRAY)
image = cv2.adaptiveThreshold(gray_image, 255, cv2.ADAPTIVE_THRESH_GAUSSIAN_C,
cv2.THRESH_BINARY, 99, 0)
```

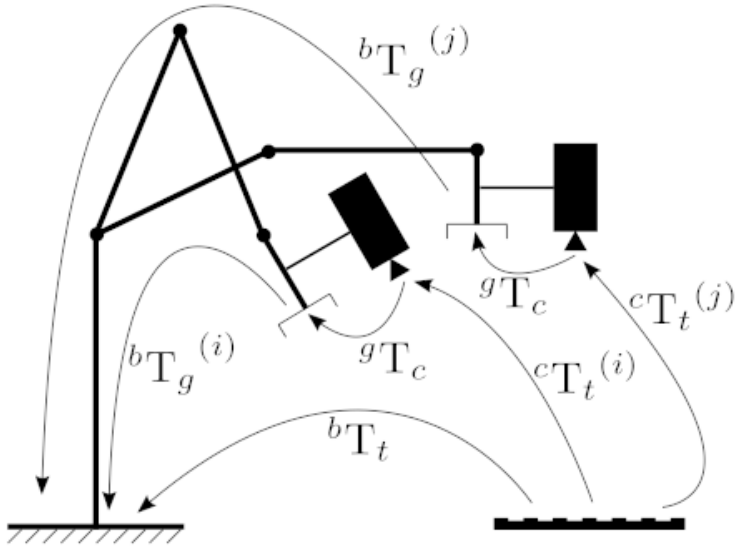
Camera Pose from Aruco Marker




```
arucoDict = cv2.aruco.getPredefinedDictionary(ARUCO_DICT[Type])
detectorParams = cv2.aruco.DetectorParameters()
detector = cv2.aruco.ArucoDetector(arucoDict, detectorParams)
corners, ids, rejected_candidates = detector.detectMarkers(image)

imagePoints = np.array(corners[0], dtype=np.double)
```

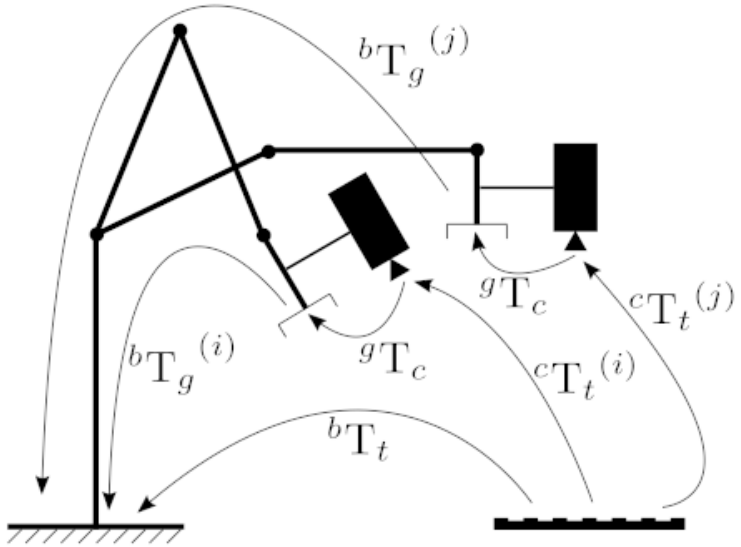
Camera Pose from Aruco Marker

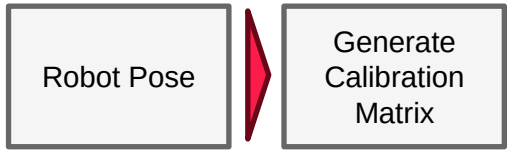



```
_, rvec, tvec = cv2.solvePnP(objPoints, imagePoints, camera_matrix,
dist_coeffs)

rot_mat, _ = cv2.Rodrigues(rvec)
camera_position = np.matrix(rot_mat).T * np.matrix(tvec)
```

Camera Pose from Aruco Marker

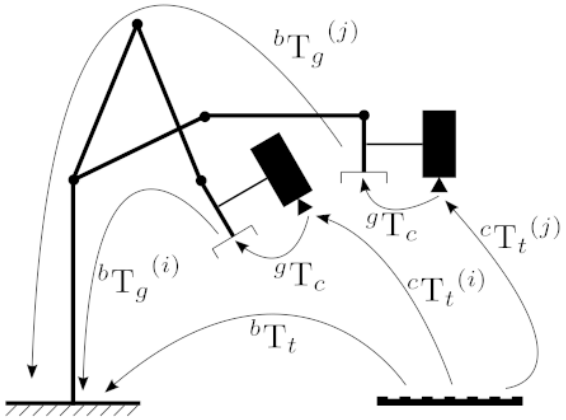


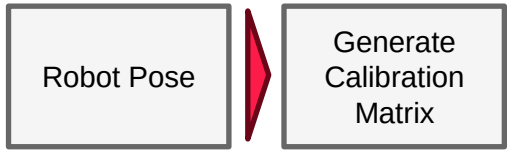



```
robot_pose = sim.getObjectPose(target_handle, base_handle)
robot_poses.append(robot_pose)

time.sleep(1)  # 시간 지연
Command.move_linear(sim, tip_handle, target_handle, cal_pose[i])
```

Camera Pose from Aruco Marker






```
    hand_eye_transformation, _ = cv2.calibrateHandEye(
        robot_poses, robot_poses,  # 첫 번째 요소는 회전 행렬, 두 번째 요소는 위치 벡터
        camera_poses, camera_poses,  # 동일하게 첫 번째 요소는 회전 행렬, 두 번째 요소는 위치 벡터
        method=cv2.CALIB_HAND_EYE_TSAI  # Tsai 알고리즘을 사용
    )
```