UGRP Seminar

Devising a Visual Pose Correction for robotic arm performing sheet metal forming

G R Yogesh

Guided By:

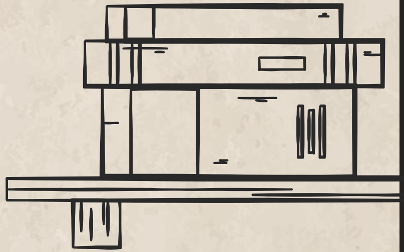
Prof. Dr. Hariharan

Prof. Anuj Kumar Tiwari

01

Understanding the Problem

Need of Visual Feedback for collaborative robots



Need of Visual/Non-Contact Feedback

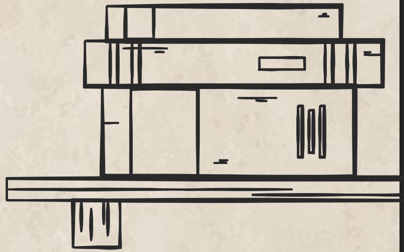
- Robotic sheet metal forming is done using two robots where one acts as a active robot and forms the sheet whereas other robot will give support from the backside so that there won't be concentrated stress built up in the sheet.
- Both the robot's end effector must maintain tight tolerances in x,y direction as its pose should overlap on one another and z difference should be constant and controllable parameter
- Robot experience deflection due to uneven or unexpected force application on end-effector or due to communication lag or other reason. This error accumulates over time and affects overall performance.
- Robot's DH parameters (which is used for forward model) has manufacturing tolerances due to which there is slight variation in actual and theoretical position of end effector.
- Active correction and calibration enhances the performance of the forming

Active feedback system involves both contact and non-contact ways. We prefer non-contact method so that the observation does not interfere with the system

02

Ideation

Initial ideation generated from the problem statement



Ideation

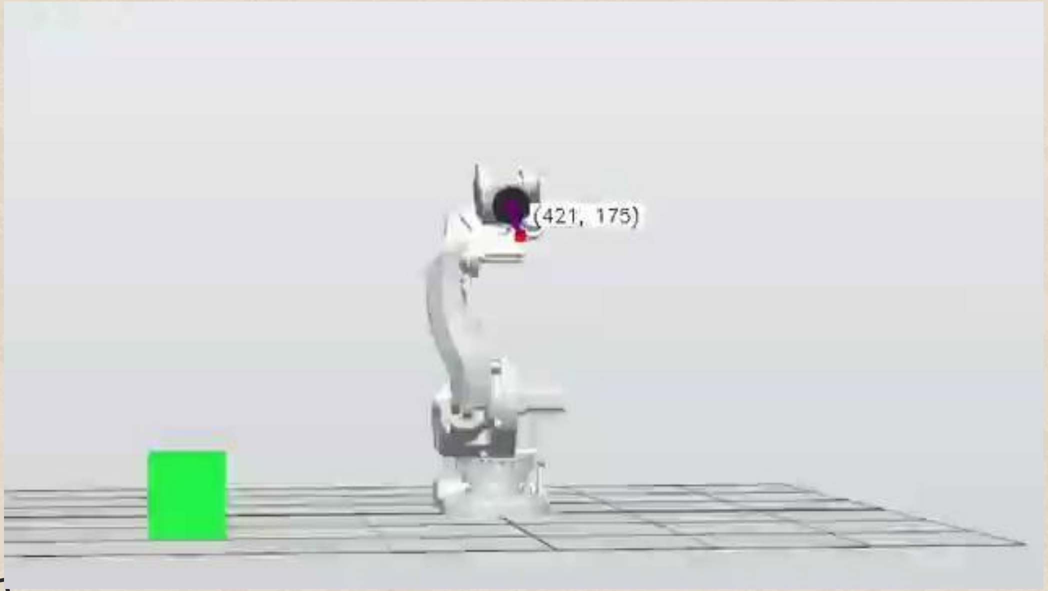
Use of Vision Based Pose Estimation:

- Use of Markers like (Aruco, Charuco, April-Tags, etc) and marker detection using computer vision algorithm.
- Using active model to segment end effector and track their pose.
- Using Grid based Imaging where we segment and track a point in end effector.

Pros and Cons of these Methods:

- Markers:
  - Pros:
    - Less computationally intensive, Best for Position+Orientation Tracking
  - Cons:
    - Lesser Accuracy (due to camera properties), Visibility Issue
- Model Based Segmentation:
  - Pros:
    - No-Visibility Constraint, Can handle complex computations
  - Cons:
    - Heavily computationally intensive, Lesser Accuracy

Model Based Pose Estimation



Specifications:

Used SAM2
Segmentation model

Observations and conclusions:

- Able to segment end effector in simple background
- Precision is in order of 1e0 mm
- Works fine for front view but might lose its efficiency in top or side view

Pros and Cons of Model Based Pose Estimation

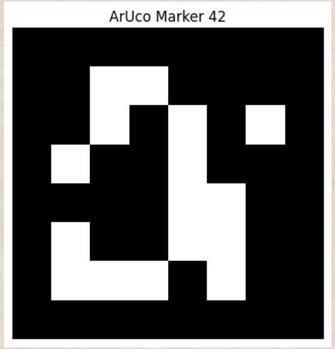
✅ Pros of Model-Based Segmentation (e.g., SAM2)

2. **Generalizable Across Domains** – Trained on diverse datasets, SAM2 handles unseen categories without retraining.
3. **Supports Zero-Shot and Prompt-Based Input** – It works with points, boxes, or masks as prompts, giving flexible control.
4. **Excellent Object Boundary Visibility** – SAM2 maintains crisp boundaries, even for thin or irregular-shaped objects.

❌ Cons of Model-Based Segmentation (e.g., SAM2)

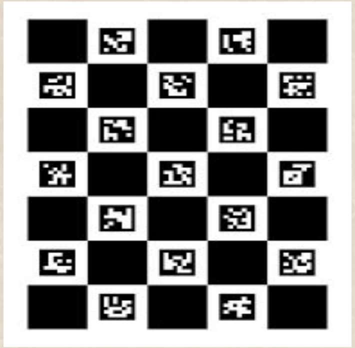
1. **False Triggers in Cluttered Scenes** – It can misfire on background textures or overlapping objects, especially in complex environments.
2. **Lower Precision on Occluded or Transparent Objects** – Performance drops sharply when the target is partially hidden or visually ambiguous.
3. **Requires High-Quality Input** – Blurry, low-light, or distorted images reduce its segmentation accuracy significantly.
4. **Heavy Resource Requirement** – Needs substantial VRAM and compute power for high-res inference or batch processing.

Marker Based Pose Estimation



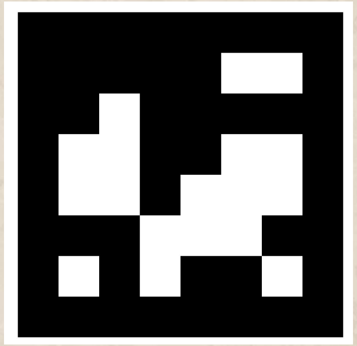
Aruco Code

A type of binary square fiducial marker used for pose estimation and camera calibration, commonly supported in OpenCV



Charuco Code

A hybrid marker system combining ArUco markers and a chessboard grid for more accurate and robust corner detection.



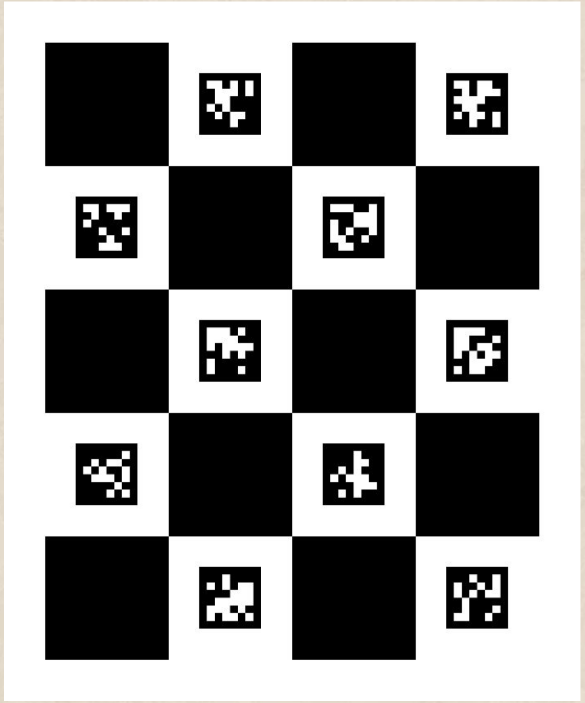
April Tag

A highly robust visual fiducial system designed for reliable detection and pose estimation in noisy or distorted environments.

Comparison between tags

Criteria	ArUco	ChArUco	AprilTag
Precision	Moderate – limited by resolution and marker size	High – subpixel accuracy using chessboard corners	High – robust even at lower resolutions
Detection Robustness	Moderate – works well in clean conditions	High – performs well under moderate occlusion	Very High – strong against noise and motion blur
Lighting Sensitivity	Sensitive to poor or uneven lighting	Better – high contrast chessboard helps detection	Excellent – designed for variable lighting conditions
Distortion Handling	Limited – less robust under perspective or lens warp	Moderate – better than ArUco due to corner interpolation	Excellent – resilient to perspective and radial distortion
Marker Complexity	Low – simple binary pattern, easy to generate	Medium – requires full grid of markers and checkerboard	High – denser encoding, more unique tag patterns
Computational Cost	Low – fast detection via OpenCV	Medium – subpixel interpolation adds processing	High – slower but optimized in latest implementations
False Positives	Moderate – higher risk with smaller markers or noise	Low – checkerboard reduces misidentification	Very Low – high Hamming distance minimizes errors
Partial Occlusion	Poor – fails if any corner is hidden	Good – detection possible if enough visible	Very Good – works with partial tag visibility

Proposed Strategy Chapril Board



We want Pros of both Charuco Code and April Tags while we can slightly compromise computations:

Charuco – aruco + april = Chapril Board

Explanation:

We replaced all aruco tags in charuco board and added april tags.

This will give us more reliable localization and better orientational estimation

Pros and Cons of this proposed strategy

✅ Pros

1. **Better Detection Accuracy** – AprilTags offer superior robustness to noise, blur, and lighting variations compared to ArUco markers.

2. **Fewer False Positives** – AprilTags reduce misidentification due to stronger internal error detection.

3. **Improved Orientation Decoding** – AprilTags resolve orientation ambiguities more reliably than ArUco markers.

❌ Cons

5. **Potentially More CPU Intensive** – AprilTag detection can be more computationally demanding than ArUco.

6. **Incompatible with OpenCV Calibration** – OpenCV’s ChArUco calibration tools only support ArUco-based boards.

7. **Requires Custom Pipeline** – You must build your own detection and calibration process from scratch.

Initial Ideation

To devise a novel vision based feedback system which does the following:

- **Sensor Data for Pose Estimation**: Combine checkerboard and AprilTag or only AprilTag detection to compute high-accuracy end-effector pose measurements via OpenCV/ArUco libraries, reducing occlusion-related errors.

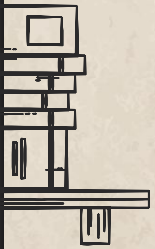
- **Sensor Fusion of Model data and Vision data:** Model data's (dh parameters) errors and Vision data's (accuracy) errors are minimized by sensor fusing datas using Unscented Kalman Filtering Algorithm till Kth time step convergence

- **Iterative DH Parameter Calibration**: Compare the robot's forward kinematic (DH model) predictions with vision-sensor data over *k* iterations, then apply Scipy's Levenberg-Marquardt optimization to refine DH parameters by minimizing pose residuals.

- **Real-Time Trajectory Correction**: Deploy a feedback loop where vision-detected pose deviations trigger joint-angle adjustments using inverse kinematics, ensuring trajectory compliance during operation.

- **ABB RAPID-PC Interface Integration**: Establish communication via RobotStudio SDK or TCP/IP sockets to dynamically update joint targets in the robot's RAPID program during runtime.

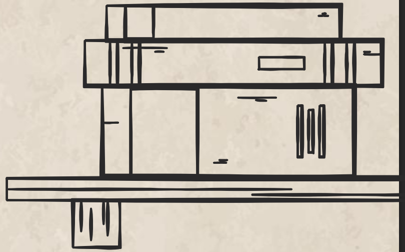
- **Convergence Validation**: Monitor error thresholds (e.g., positional ±1 mm, angular ±0.5°) between model predictions and sensor data to terminate calibration and activate real-time correction.



03

Literature Review

Literature review to analyse different methodologies



Literature Review

1) A novel vision-based calibration framework for industrial robotic manipulators

Doi: https://doi.org/10.1016/j.rcim.2021.102248

- The paper presents a vision-based calibration framework for improving the accuracy of industrial robotic manipulators. The authors specifically deal with compensating **geometric and non-geometric errors** in robot kinematics using **external visual feedback**, such as from **AprilTags and checkerboard patterns**.

- The robot moves through predefined configurations where visual pose errors from AprilTags are computed and fed into a Levenberg–Marquardt optimizer to iteratively update the DH parameters. This loop refines the kinematic model for improved accuracy.

- A 3D-printed **Rhombicuboctahedron structure** is placed on the robot's end-effector, with ArUco markers attached on its surfaces. The camera captures this structure, and ArUco markers are detected using image processing (thresholding, contour detection, ID decoding, and corner refinement).

- The camera is first calibrated using a **ChArUco board** to obtain intrinsic (focal length, principal point) and distortion parameters. This ensures accurate 2D-3D mapping.

- Using the known 3D positions of the marker corners (from the 3D model) and their 2D image locations, the **Perspective-n-Point (PnP)** algorithm computes the **pose of the camera** relative to the marker structure (called RAM – Rhombicuboctahedron ArUco Map).

- Using pre-calibrated **hand–eye transformation matrices**, the pose of the **end-effector in the robot base frame** is computed from the camera-to-marker pose.

Literature Review

2) A novel vision-based calibration framework for industrial robotic manipulators

Doi: https://doi.org/10.1016/j.precisioneng.2024.08.010

- The paper presents a vision-based calibration framework to improve the absolute positioning accuracy of industrial robots using a monocular camera and ArUco marker maps. It addresses the limitations of traditional calibration methods restricted by the camera's field of view, enabling large-scale workspace calibration.
- The robot collects visual data by moving through various configurations, where end-effector poses are estimated using ArUco markers and refined iteratively using the Levenberg–Marquardt algorithm to correct DH parameters. This closed-loop process reduces cumulative error and enhances the robot's kinematic model.
- An eye-in-hand camera mounted on the robot captures multiple ArUco markers with unique IDs placed across the workspace. These markers are used to form a globally optimized ArUco map through overlapping observations and loop closures.
- The camera is calibrated using a checkerboard to determine intrinsic parameters. Then, using 3D marker coordinates and their 2D image projections, the Perspective-n-Point (PnP) algorithm estimates the pose of the camera relative to the ArUco map.
- With known hand–eye transformation parameters, the pose of the robot's end-effector in the base frame is computed from the camera pose, enabling precise error estimation and correction.
- To model the error, the authors use a differential kinematic approach based on the Modified Denavit-Hartenberg (MDH) model. They derive a Jacobian that relates small deviations in DH parameters ($\Delta\alpha$, Δa, Δd, $\Delta\theta$, $\Delta\beta$) to pose errors in position and orientation. These deviations are iteratively minimized using the LM algorithm until the kinematic model aligns closely with the measured poses.
- This paper claims to have better efficiency in comparison, traditional checkerboard-based calibration reduced the error only to **0.722 mm**. While both methods improved accuracy, the ArUco map approach showed **notably better performance**, especially across a **larger calibration space**, due to its robustness against field-of-view limitations and support for distributed marker tracking.

Literature Review

3) Searching for an Accurate Robot Calibration via Improved Levenberg-Marquardt and Radial Basis Function System

Doi: 10.1109/ICRA.2016.7487606

- The paper presents a hybrid calibration framework to improve the positioning accuracy of industrial robots by addressing both **kinematic and dynamic errors**. It proposes an enhanced Levenberg–Marquardt (LM) algorithm integrated with a **Fuzzy Proportional-Integral-Derivative (FPID)** controller and a **Radial Basis Function Neural Network (RBFNN)**, forming a two-stage system called **FPLM-RBFNN**.
- The robot is modeled using standard Denavit–Hartenberg (DH) parameters, and the pose errors are defined as the difference between the measured and predicted end-effector positions. These errors are mapped to parameter errors through a Jacobian, and an LM-based optimization is applied to iteratively minimize them. The FPID controller adaptively tunes the LM updates using fuzzy logic to improve convergence speed and accuracy.
- The second stage of the calibration uses an RBF neural network to model and compensate for **dynamic errors**, which are harder to capture analytically. The network uses six hidden neurons corresponding to six joints and adjusts its weights through gradient descent based on the difference between predicted and actual end-effector positions.
- This hybrid system was tested on multiple datasets including ABB IRB120 and HSR JR680 robots. On the HSR robot dataset, the proposed method reduced RMSE from 5.19 mm to **0.32 mm**, outperforming all other algorithms including standalone LM, PSO, GA, and EKF. Compared to plain LM (0.68 mm RMSE), it achieved a **52.94% improvement in accuracy** and a **66.30% reduction in computation time** when using FPID-tuned LM alone.
- This paper is particularly important to our project because it shows how **sensor-model fusion and intelligent optimization (LM + neural networks)** can be combined to compensate for both **systematic (DH-based) and complex (dynamic) errors**. It highlights the use of visual or indirect measurement data to refine model predictions and demonstrates a scalable path to high-accuracy robot calibration without relying on expensive motion capture systems.

Literature Review

4) Unscented Kalman Filter and 3D Vision to Improve Cable Driven Surgical Robot Joint Angle Estimation

Doi: https://doi.org/10.1016/j.rcim.2021.102248

- The paper presents a sensor fusion framework to enhance joint angle estimation for cable-driven surgical robots, using low-cost stereo vision combined with an **Unscented Kalman Filter (UKF)**. It addresses the challenges of estimating joint angles accurately in cable-driven systems, where **motor-mounted encoders alone are insufficient** due to nonlinearities like cable stretch and low stiffness.
- The robot's end-effector is tracked using a passive color marker and a stereo camera system built from two low-cost webcams. The 3D position of the marker is estimated via disparity from stereo images, and a **region-of-interest (ROI)** based detection method is used to improve marker localization performance under varying lighting conditions. The 3D pose is transformed to the robot base frame using known transformations computed at the mechanical hard-stop position.
- A **dual Unscented Kalman Filter (UKF)** framework is implemented, where one UKF estimates the system states (joint angles and velocities), and the other estimates kinematic and dynamic parameters (such as pulley radius, cable stiffness, and damping). The UKF operates on a reduced-order model where motor states are known from encoders, so only joint angle states are estimated. Observations from stereo vision provide noisy joint measurements, which are fused into the UKF to correct the predicted state. The state update is governed by a nonlinear dynamic model of the robot, and the UKF uses deterministic sampling (sigma points) to propagate uncertainty through the nonlinearity. For parameter estimation, the UKF assumes known states while tuning parameters, and alternates with the state estimator. This structure enables simultaneous learning of system dynamics and correction of joint angle estimates in real time.
- This paper is important to our project as it demonstrates how **fusing vision and model-based prediction** with a **filtering framework (UKF)** enables accurate state estimation, even under noisy or partial observations. It also validates the idea of using low-cost cameras and vision-based error correction in real-time robotic applications, similar to our proposed vision-based correction system for 6-DOF industrial arms.

Key Takeaways

1. Sensor Data for Pose Estimation **Papers**: *Vision-based calibration framework*, *ArUco map-based calibration*

Summary:

Vision-based pose estimation using AprilTags/ArUco provides accurate 6D measurements.

Combining checkerboards and tags enhances robustness to occlusion and expands the effective workspace.

2. Sensor Fusion of Model and Vision Data **Papers**: *Sensor fusion for joint angle estimation*, *Improved Levenberg–Marquardt calibration*

Summary:

UKF fuses noisy vision data with kinematic predictions for accurate joint estimation.

This fusion handles nonlinearity and noise, enabling robust recursive parameter correction.

3. Iterative DH Parameter Calibration **Papers**: *Vision-based calibration framework*, *ArUco map calibration*, *Improved LM calibration*

Summary:

Levenberg–Marquardt minimizes pose errors by refining DH parameters iteratively.

Jacobian-based modeling captures the impact of small DH deviations on end-effector pose.

4. Real-Time Trajectory Correction **Papers**: *Sensor fusion*, *Improved LM calibration*

Summary:

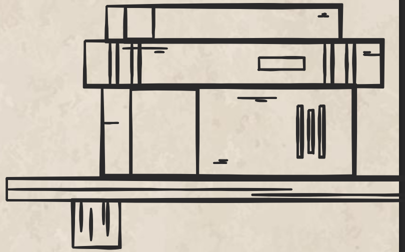
Vision feedback triggers real-time corrections via inverse kinematics for trajectory tracking.

Sub-millimeter accuracy is achieved by continuously adjusting joint commands based on sensor data.

04

Refined Ideation

Refinement of ideation after literature analysis



Refined Ideation

- **Sensor Data for Pose Estimation**
  Combine checkerboard and AprilTag (or only AprilTag) detection to compute high-accuracy end-effector pose measurements using OpenCV/ArUco libraries. This helps reduce occlusion-related errors and improves spatial coverage.

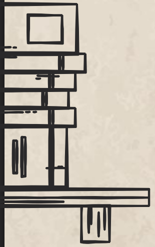
- **Sensor Fusion of Model Data and Vision Data**
  Fuse model-based predictions (from DH parameters) and vision-based measurements using the Unscented Kalman Filter (UKF). This sensor fusion minimizes individual sensor errors and continues iteratively until convergence at the k^{th} time step.

- **Iterative DH Parameter Calibration**
  Compare predicted end-effector poses from forward kinematics with vision-based measurements over multiple iterations. Apply the Levenberg–Marquardt algorithm (using SciPy) to iteratively update DH parameters by minimizing the residual pose errors.

- **Real-Time Trajectory Correction**
  Use real-time feedback from vision-based pose estimation to detect deviations and apply joint-angle corrections using inverse kinematics. This maintains trajectory adherence even under dynamic changes or modeling inaccuracies.

- **ABB RAPID-PC Interface Integration**
  Set up communication between the ABB robot and a PC using the RobotStudio SDK or TCP/IP sockets. This allows for real-time updates to joint targets in the robot's RAPID program during execution.

- **Convergence Validation**
  Monitor pose errors continuously, and when thresholds such as positional error ≤ ±1 mm and angular error ≤ ±0.5° are achieved, terminate the calibration phase and switch to real-time correction.



WorkFlow

- **Sensor Data for Pose Estimation**
  Combine checkerboard and AprilTag (or only AprilTag) detection to compute high-accuracy end-effector pose measurements using OpenCV/ArUco libraries. This helps reduce occlusion-related errors and improves spatial coverage.

Steps followed:

1. Camera Calibration (One-Time Step)
Use multiple checkerboard images from different angles to calibrate the camera. Corners are detected with `cv2.findChessboardCorners()`, and `cv2.calibrateCamera()` provides the intrinsic matrix (`K`) and distortion coefficients (`D`).

2. Marker Detection (Per Frame)
In each frame, detect AprilTags using `cv2.aruco.detectMarkers()` and estimate their pose with `cv2.aruco.estimatePoseSingleMarkers()` using `K` and `D`. Detect the checkerboard again with `cv2.findChessboardCorners()` and estimate pose using `cv2.solvePnP()`.

3. Pose Fusion
If both markers are detected, fuse the poses using a weighted average or a filter (e.g., Kalman). If only one is detected, use the one with better visibility or lower reprojection error.

4. End-Effector Pose Computation
Transform the fused pose to the robot or world frame using a known static transformation between the marker and the end-effector.

5. Visualization
Visualize the estimated pose by drawing coordinate axes on the image using `cv2.drawFrameAxes()` for both AprilTag and checkerboard detections.

WorkFlow

- **Sensor Fusion of Model Data and Vision Data**
  Fuse model-based predictions (from DH parameters) and vision-based measurements using the Unscented Kalman Filter (UKF). This sensor fusion minimizes individual sensor errors and continues iteratively until convergence at the k^{th} time step.

Steps followed:

1. Model Initialization (One-Time Step)
The UKF is initialized with 6D state and measurement spaces using Merwe sigma points. A `Transforms` model is built from DH parameters and initial transform `ini_T`. Initial state and uncertainty are set.

2. State Prediction (fx Function)
The `fx` function predicts the next pose using the model Jacobian and input joint velocities. If control input `u` is provided, the pose is updated using $x + J(q) \cdot \dot{q}$; otherwise, it remains the same.

3. Measurement Model (hx Function)
The `hx` function returns the current state directly, assuming that vision measurements match the state variables (e.g., 6D pose).

4. Sensor Fusion Loop (Per Time Step)
At each step, `predict()` updates the state from the model, and `update(z)` corrects it using vision-based measurements. This fusion reduces noise and improves accuracy.

5. Output Retrieval
The current pose estimate is accessed using `get_x()`, which returns the UKF's best estimate after combining both model and sensor data.

WorkFlow

- **Iterative DH Parameter Calibration**
  Compare predicted end-effector poses from forward kinematics with vision-based measurements over multiple iterations. Apply the Levenberg–Marquardt algorithm (using SciPy) to iteratively update DH parameters by minimizing the residual pose errors.

Steps followed:

1. Initialize DH Parameters
Start with initial guesses for the DH parameters of the robot. These parameters define the robot's kinematic model.

2. Forward Kinematics Prediction
Use the current DH parameters to compute the predicted end-effector pose via forward kinematics. This is done for each iteration based on the current parameters.

3. Vision-Based Measurements
Capture the real end-effector pose using vision-based measurements (e.g., from a camera or marker-based system).

4. Residual Error Calculation
Calculate the residual errors by comparing the predicted pose (from DH parameters) and the vision-based measurement at each iteration. The error is typically the difference in position and orientation.

5. Levenberg–Marquardt Optimization
Apply the Levenberg–Marquardt algorithm from SciPy to iteratively update the DH parameters. The goal is to minimize the residual pose errors by adjusting the DH parameters through each iteration.

6. Convergence
Repeat the process until the residual errors converge to a minimum, indicating that the DH parameters have been calibrated.

WorkFlow

- **ABB RAPID-PC Interface Integration**
  Set up communication between the ABB robot and a PC using the RobotStudio SDK or TCP/IP sockets. This allows for real-time updates to joint targets in the robot's RAPID program during execution.

Steps followed:

1. **Set up Communication**
   Establish communication between the ABB robot and a PC using TCP/IP sockets or the RobotStudio SDK. This setup allows real-time updates to joint targets in the robot's RAPID program during execution.
2. **RobotStudio Server (RAPID Code)**
   The RAPID program runs a server on the robot controller that listens for incoming connections. It receives joint position data and updates the robot's position in real-time.
3. **Data Exchange**
   The PC sends joint position data (e.g., position coordinates, orientation) to the robot via the established TCP/IP connection. The robot, using the RAPID program, parses this data, updates the joint targets, and moves the robot accordingly.
4. **Socket Communication (PC Side)**
   The PC continuously sends data packets to the robot via the TCP/IP socket connection. It formats the joint positions into strings, sends them over the network, and waits for acknowledgment from the robot.
5. **Robot Response Handling**
   After receiving and processing the data, the robot sends a response (e.g., confirmation or updated position). The PC listens for this response to ensure that the data exchange was successful.
6. **Connection Management**
   The communication loop continues for continuous control, with periodic updates being sent to ensure smooth robot operation. Upon completion, the connection is closed.

Summary

Project Overview – Vision-Based Pose Correction for 6-DOF ABB Robotic Arm

Objective:

- Develop a vision-based correction system to improve pose accuracy of a 6-DOF ABB robotic arm during sheet metal forming using checkerboard + AprilTag fusion.

Motivation:

- Precise end-effector alignment is critical in collaborative tasks.
- Pose errors arise due to mechanical tolerances and dynamic forces.
- Traditional model-based calibration suffers from sensor noise and inaccuracies.

Key Innovations:

- Combines AprilTag and checkerboard-based pose estimation.
- Uses Unscented Kalman Filter (UKF) for sensor fusion.
- Calibrates DH parameters via Levenberg-Marquardt optimization.

Summary:
This project proposes a hybrid vision-kinematics correction system using UKF and iterative DH calibration, achieving improved pose accuracy and reliability for industrial robotic applications.

Summary

Workflow & System Architecture

Workflow Steps:

- **Camera Calibration:** One-time intrinsic calibration with checkerboard.
- **Pose Estimation:** Real-time AprilTag/checkerboard detection for 6D pose.
- **Sensor Fusion:** UKF merges vision and model-based estimates.
- **DH Calibration:** LM optimization refines DH parameters iteratively.
- **Real-Time Correction:** Visual errors trigger trajectory updates via IK.
- **Communication:** PC sends corrections to ABB via TCP/IP or RobotStudio SDK.

Expected Outcomes:

- Sub-millimeter positional accuracy.
- Adaptive, non-contact correction in real-time.
- Scalable framework for advanced robot calibration and control.

Summary:
A modular pipeline integrating vision and model-based control ensures accurate pose correction. The system enhances adaptability and precision for dynamic industrial tasks.

Thanks!

I would love to hear your feedback

Acknowledgment:

I would like to express my sincere gratitude to
Prof. Dr. Hariharan,
Prof. Anuj Kumar Tiwari, and
Prof. Sourav Rakshit

for serving on the evaluation panel and for their valuable time, insights, and feedback throughout the evaluation of this project.