Vision-Guided Robotic Manipulation: A Computer Vision-Integrated Robotic Arm for Real-World Object Interaction

By, **Eyes of AI**
Arnab Singha , Sudam Kumar Paul

Ramakrishna Mission Vivekananda Educational and Research Institute
Belur, G. T. Road, PO Belur Math, Howrah, West Bengal 711202

April 30, 2025

1 Introduction

1.1 Background

Computer vision-guided robotic arms represent a significant advancement in the field of automation and robotics. By integrating camera-based perception and image processing techniques, these systems enable robots to perceive, understand, and interact with their environments more intelligently. Through object detection, recognition, and localization, robotic arms equipped with vision systems can perform complex manipulation tasks with greater flexibility and accuracy. This combination of vision and robotic control expands the applicability of robots to dynamic, unstructured, and unpredictable real-world settings, ranging from industrial automation to service robotics.

1.2 Motivation

The growing demand for adaptable and autonomous robotic systems across various industries has driven interest in computer vision-guided manipulation. Traditional robots, often limited to repetitive tasks in controlled environments, lack the perception capabilities required for dynamic operations. By empowering robotic arms with vision-based intelligence, it becomes possible to tackle a wider variety of tasks without the need for extensive reprogramming or environmental constraints. This approach opens pathways toward more efficient, cost-effective, and intelligent automation solutions, motivating research and development in this promising area.

1.3 Objectives

- Detect objects in a specified workspace of the robotic arm (serial manipulator) using computer vision techniques.
- Identify it's actual location on real world w.r.t. a defined coordinate system.
- Reach the robot's end effector to the location.
- Manipulate the object with the robots end effector.
- Pick up an object from anywhere in the workspace and place in a fixed location within the robot's range.

2 Literature Review

In 2022, a review titled *"A Comprehensive Review of Vision-Based Robotic Applications: Current State, Components, Approaches, Barriers, and Potential Solutions"* by Md Tanzil Shahria, Md Samiul Haque Sunny, Md Ishrak Islam Zarif, Jawhar Ghommam, Sheikh Iqbal Ahamed, and Mohammad H Rahman [1] presents an extensive examination of vision-based robotic manipulation systems and their evolution over recent years. The study systematically analyzes 46 significant works published between 2016 and 2022, focusing on integrating computer vision, machine learning, and control algorithms to enhance robotic perception and interaction.

The research outlines critical methodologies, including the adoption of sensory devices like RGB and depth cameras, the application of deep learning models such as CNNs, RNNs, and DNNs for visual data processing, and the use of real-time experiments alongside simulated platforms for performance evaluation. Specific approaches like visual servoing, reinforcement learning, and vision-based grasping techniques are surveyed, highlighting how learning-based models increasingly outperform traditional control methods. Challenges such as real-world generalization, computational delay, sensor limitations, and robustness to dynamic environments are emphasized, with potential solutions including the adoption of multi-modal sensing and lightweight neural networks.

The paper concludes that vision-based robotic manipulation has made significant strides, establishing deep learning-integrated vision systems as a cornerstone for future autonomous robotics while recognizing the necessity for more robust, real-world deployment to bridge the gap between laboratory success and practical application.

In 2020, a study titled *"Robot Vision for Manipulation: A Trip to Real-World Applications"* by Ester Martinez-Martin and Angel P. Del Pobil [2] presents an in-depth exploration of the evolution of vision-guided robotic manipulation from controlled industrial environments to real-world dynamic settings. The study analyzes various vision-based robotic developments, focusing on the challenges encountered and solutions proposed to enable autonomous robot operation across structured, semi-structured, and unstructured environments.

The research outlines methodologies including the use of fisheye cameras for wide-area monitoring, motion detection through adaptive background modeling, and object/person tracking to enable safe human-robot collaboration in industrial settings. It further explores traditional computer vision methods, such as Hough transforms and OCR for book identification in libraries, and deep learning models like ResNet and SIFT for object recognition tasks in warehouses. For real-world assistive robots, the study proposes biologically inspired approaches combining color, motion, and shape analysis to achieve fast, accurate object detection and grasp supervision without relying heavily on deep learning models.

Results demonstrate that the proposed vision systems achieve high performance, with motion detection accuracy exceeding 97% and robust object recognition in cluttered and dynamic environments. The study emphasizes that no single vision solution fits all tasks and environments, highlighting the importance of combining classical computer vision techniques, deep learning methods, and biologically inspired models to overcome environmental challenges and ensure robust robotic manipulation in real-world applications.

In 2024, a study titled *"Vision-Based Object Manipulation for Activities of Daily Living Assistance Using Assistive Robot"* by Md Tanzil Shahria, Jawhar Ghommam, Raouf Fareh, and Mohammad Habibur Rahman [3] presents a comprehensive exploration of semi-autonomous vision-based robotic systems designed to assist individuals with upper and lower extremity dysfunctions in performing daily activities.

The study focuses on developing a pick-and-place robotic system combining deep learning-based object detection and real-time 3D localization using a depth camera to enable independent interaction with everyday objects.

The research outlines methodologies including the use of a 6-DoF robotic manipulator (xArm6), integration of an Intel RealSense D435 depth camera, and training a YOLOv5s model on a customized ADL object dataset. Techniques such as real-time depth-based localization, inverse kinematics, and distance-controlled robotic movement are employed to achieve accurate object grasping. Results demonstrate that the semi-autonomous system achieves a 72.9% overall task success rate in real-world pick-and-place experiments, significantly reducing task completion time compared to manual joystick control. Challenges related to grasping complex-shaped objects and depth perception limitations are identified, with future work proposed to improve grasping strategies, expand the ADL dataset, and integrate the system into wheelchair platforms for enhanced user accessibility.

This work emphasizes the potential of combining deep learning, real-time perception, and adaptive control to create effective assistive robotic systems that promote greater independence for individuals with physical impairments.

In 2025, a study titled *"Autonomous Object Tracking with Vision-Based Control Using a 2-DOF Robotic Arm"* by Umesh Kumar Sahu, Mebin K. S., Abhinav K., Muhammed Muzammil P, Ankur Jaiswal, Umesh Kumar Yadav, and Santanu Kumar Dash [4] presents a focused exploration of integrating vision-based control with robotic arm tracking in dynamic environments. The study proposes a lightweight and cost-effective approach, combining deep learning for feature extraction with an image-based visual servoing (IBVS) control strategy to enable autonomous tracking using a 2-DOF robotic arm.

The research outlines methodologies including real-time 2D feature detection using MediaPipe deep learning models, camera calibration for 3D coordinate extraction, and the application of IBVS for minimizing the visual error between target and robotic end-effector. The system architecture integrates OpenCV for image processing, Arduino-based control for the robotic arm, and CoppeliaSim for initial simulation validation. Results demonstrate that the developed system achieves robust real-time tracking performance both in simulation and in hardware experiments, maintaining stability even under dynamic target movements. Challenges such as field-of-view loss and controller gain sensitivity are acknowledged, and future work is directed toward integrating reinforcement learning to improve system adaptability.

This work emphasizes the effectiveness of combining deep learning-driven perception with classical control theory to build efficient, real-time vision-guided robotic systems using minimal computational resources.

3 Methodology

3.1 Architecture

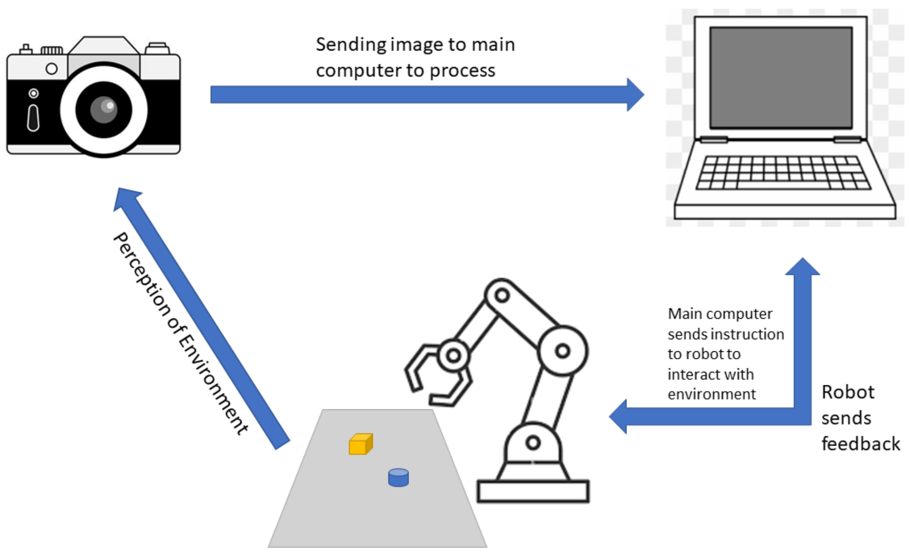
Sending image to main computer to process
Perception of Environment
Main computer sends instruction to robot to interact with environment
Robot sends feedback

Figure 1: Architecture of Robotic Arm System.

- There is a robot and a surface where some different shaped and colored objects are kept within it's range.
- Now mobile camera will capture the image of the surface containing objects.
- Then it will send the captured image to main computer to process. Then computer will detect the objects and their centre points.
- Through WiFi the computer will send the location and commands to the robot for necessary action.

- By using inverse kinematics robot will change it's angles and reach to the objects to grip / release it.

3.2 Mechanical Description of Robot

The robot is a serial manipulator. The robot's base is a circular disk with radius 6.6cm. The robot is fixed at the center of the base. It has 3 links (L1, L2 and L3) and three rotational joints(base, shoulder, elbow) and an end effector(gripper). The three links are of length L1 = 9.5 cm , L2 = 7 cm ,L3 = 10.6 cm . The rotational joint at the base is for rotating the robot horizontally, i.e. about it's own axis. The rotational joints of shoulder and elbow are for rotating vertically and reach the desired point. All the joints are implemented using micro servo motors. All servos are capable of rotating a half circle. The gripper is controlled using a micro servo motor of similiar kind.

3.3 Inverse Kinematics For Robot

Let's consider a given arbitrary known coordinate as (x, y). Now, the robot's end effector has to reach the target coordinate (x, y). Let's consider the lengths of three links of the robot: L_1, L_2, and L_3, respectively. We can measure those lengths by scale, say $AD = d$, $AB = c$, and $BC = a$. Now the baseline's length is $DC = b = \sqrt{x^2 + y^2}$.

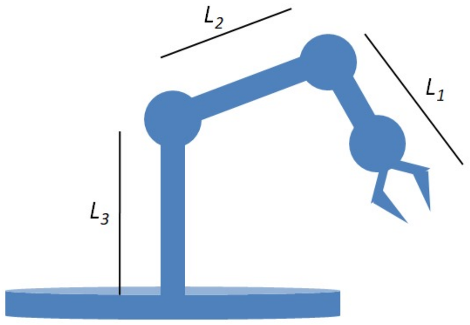

Figure 2: Robot Model

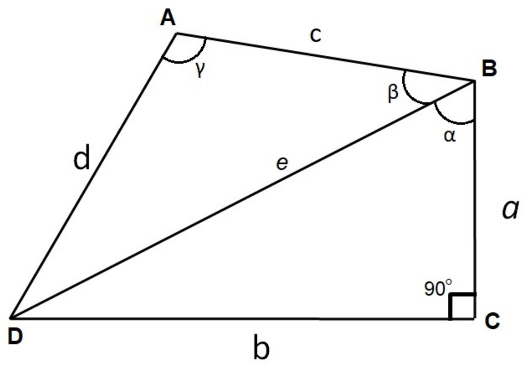

Figure 3: Formulation of Inverse Kinematics for Robot Model

Now $BD = e = \sqrt{a^2 + b^2}$, $\angle ABC = \alpha + \beta$, and $\angle BCD = 90°$.

Let $\angle BAD = \gamma$.
Now our target is to find α, β, and γ. Now by the law of cosines we get,

$$d^2 = c^2 + e^2 - 2ce \cdot \cos(\beta) \tag{1}$$
$$e^2 = c^2 + d^2 - 2cd \cdot \cos(\gamma) \tag{2}$$
$$\tan(\alpha) = \frac{b}{a} \tag{3}$$

Then by (3),

$$(\alpha) + (\beta) = \tan^{-1}(\frac{b}{a}) + \cos^{-1}\left(\frac{c^2 + e^2 - d^2}{2ce}\right) \tag{4}$$
$$(\gamma) = \cos^{-1}\left(\frac{c^2 + d^2 - e^2}{2dc}\right) \tag{5}$$

Now we will find the base angle $\angle PQR = \delta$

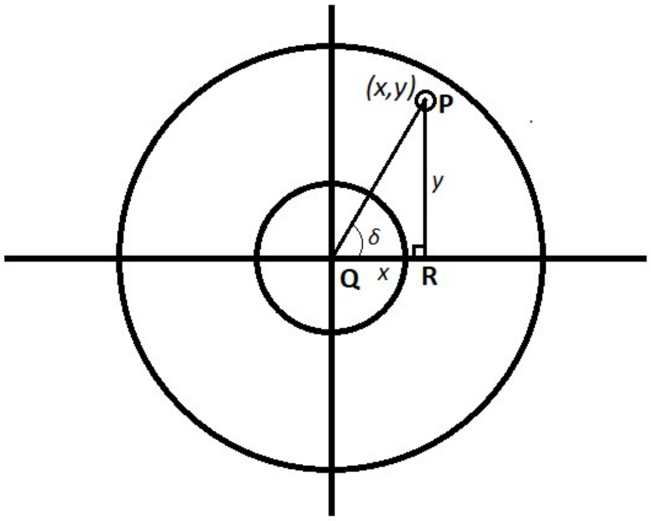

Figure 4: Triangle PQR showing the base angle δ.

From triangle PQR, we get

$$\tan(\delta) = \frac{y}{x} \tag{6}$$

From (6) we have,

$$(\delta) = \tan^{-1}(\frac{y}{x}) \tag{7}$$

3.4 Object Detection

We have implemented objects detection using contour detection methods which is crucial for various computer vision tasks such as shape analysis, object recognition, and image segmentation. The theoretical process of contour detection can be broken down into several key steps.

- **Image Preprocessing:**
  1. **Grayscale Conversion**: The first step is to convert the input image (which may be colored) into a grayscale image. This simplifies the image, reducing it to shades of gray representing the intensity of light. This step eliminates the color information, making it easier to focus on the shape and structure of the objects.
  2. **Thresholding:** After converting the image to grayscale, the next step is usually thresholding. Thresholding involves converting the grayscale image into a binary image (black and white), where all pixels above a certain intensity value are set to white (foreground), and those below are set to black (background). This creates a clear distinction between the objects (foreground) and the background, which makes contour detection easier.
- **Contour Tracing (Finding Contours):**
  1. Contour detection algorithms trace the connected components of the foreground pixels in the binary image. The process involves starting from any foreground pixel (white pixel), following the boundary (pixels that form the object outline), and recording the sequence of pixels that form the boundary. These pixels form the contour.
  2. The key idea is to trace along the boundary and return the sequence of points (x, y coordinates) that make up the contour.
  3. **Contour Retrieval Modes**: Contours can be retrieved in various modes, such as,
     - **External Contours**: Only the outermost boundary of the object is traced.

  - **All Contours**: All boundaries, including inner contours (holes), are traced.
  - **Hierarchy of Contours**: For images where objects may have holes inside them, the algorithm organizes contours into a tree-like structure where the outer contour is the parent, and inner contours (holes) are considered children.

- **Contour Analysis**: After detecting contours, finding the bounding box and the center point (centroid) of a contour are common tasks in object localization and analysis.

  1. **Bounding Box of an object**: A bounding box is the smallest rectangle that can completely enclose the contour. It is aligned with the axes (i.e., not rotated).

     Mathematical Definition:

     Given a contour, the bounding box is represented by:

     - **Top-left corner** (x, y): Coordinates of the upper-left corner of the rectangle.
     - **Width and Height** (w, h): Dimensions of the rectangle.

     For a contour, the bounding box can be defined by the following formulas:

     - **Top-left corner**: The smallest x and y coordinates of the contour's points.
     - **Bottom-right corner**: The largest x and y coordinates of the contour's points.

     The width w and height h of the bounding box are calculated as:

     $$w = \max(x_{\text{coordinates}}) - \min(x_{\text{coordinates}})$$

     $$h = \max(y_{\text{coordinates}}) - \min(y_{\text{coordinates}})$$

  2. **Center Point(Centroid) of a Contour**: The centroid (or center of mass) of a contour(approximately ($\approx$) the center point of the object) is the average position of all the points that make up the contour. It is calculated using moments.

     Mathematical Definition:

The centroid of a contour is the weighted average of the coordinates of all the points in the contour. It can be computed using the first-order moments M_{10} and M_{01}, as follows:

$$c_x = \frac{M_{10}}{M_{00}}$$

$$c_y = \frac{M_{01}}{M_{00}}$$

Where:

- M_{00} is the zeroth moment (the area of the contour),
- M_{10} and M_{01} are the first-order moments with respect to the x and y coordinates, respectively.

This formula gives the (x, y) coordinates of the contour's centroid.

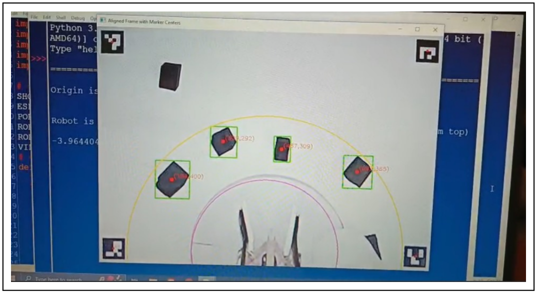

Figure 5: Object Detection

4 Implementation

4.1 Requirements

- **Software**:

  1. Windows 10, or later
  2. Python
  3. OpenCV
  4. IP Webcam
  5. Visual Studio Code

6. Arduino IDE
7. C++

- **Hardware**:

    1. Camera
    2. Laptop
    3. Arduino Uno / ESP32
    4. Servo Motors
    5. Connection Wires
    6. Power Supply
    7. Solderless BreadBoard
    8. Sticks for Chassis of Arm

4.2 Robot Hardware Setup

We are using an ESP32 micro-controller development board as the heart of our robotic arm. The 4 datalines of the 4 servo motors of the robotic arm connected to specific pins (PWM enabled) of ESP32. These pins send PWM pulses to the servo for rotating them. A +5V supply is provided to both robot and ESP32 by a power supply. Necessary GND connections are provided. The ESP32 connects with the main computer using WiFi through a common network.

4.3 Camera Setup

- There is a A4 page where 4 ArUco Markers are kept in the corners of the page.
- Computer has an reference image of the A4 page with ArUco markers.
- Camera is kept at a fixed position inclined in a 45 to 50 degree angle (in Perspective view) with the surface.
- Camera will capture the page from perspective view.
- Using reference image of page and centers of ArUco Markers, computer transforms the image in bird's eye view by homography calculation as follows.

Figure 6: Reference Image of Workspace

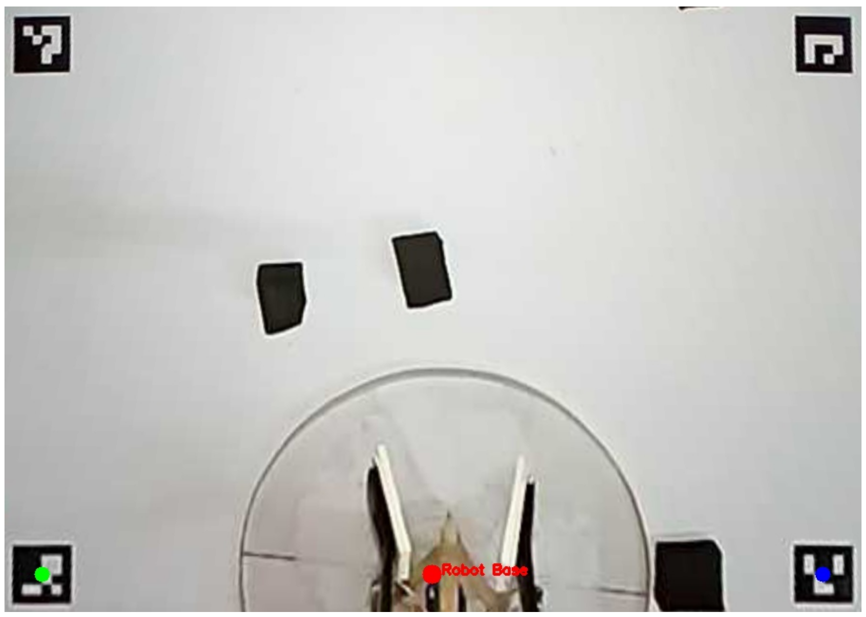

Figure 7: Registered Image of Workspace

Image Warping means transforming an image so that its shape, position, or perspective changes. When we know the corresponding points between two planes (for example, corners of a document in an image and corners of a rectangle), we can compute a special transformation called Homography to warp one view into another.

Homography Matrix: Homography is a 3×3 matrix that defines a mapping between two planes (surfaces). It mathematically relates the coordinates of points in the original image to the coordinates in the warped image.

The relationship is given by:

$$\begin{bmatrix} x' \\ y' \\ w' \end{bmatrix} = H \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

where:

- (x, y) = original point
- (x', y') = transformed (warped) point (after dividing by w')
- $H = 3 \times 3$ homography matrix

Steps for Image Warping using 4 Points:

- At first 4 centers of ArUco markers in the workspace image captured from perspective view are detected, say:

$$(x_1, y_1), \quad (x_2, y_2), \quad (x_3, y_3), \quad (x_4, y_4)$$

- These points are actually the 4 centers of ArUco markers of same ID (which are captured by the camera from perspective view) of the reference workspace image:

$$(x'_1, y'_1), \quad (x'_2, y'_2), \quad (x'_3, y'_3), \quad (x'_4, y'_4)$$

- **Calculate the Homography Matrix (H)**: A system of linear equations to find 8 unknowns is solved (since H has 9 elements but it is scale-invariant, one element such as $h_{33} = 1$ is fixed).

  The general form of the equations for each point correspondence is:

$$x'(h_{11}x + h_{12}y + h_{13}) = (h_{31}x + h_{32}y + 1)$$
$$y'(h_{21}x + h_{22}y + h_{23}) = (h_{31}x + h_{32}y + 1)$$

Using all 4 point correspondences, h_{ij} is solved using matrix methods such as Singular Value Decomposition (SVD).

- **Warp the Image**: Once we have the homography matrix H, we can apply it to the whole image (or just a selected region). Each pixel's new position is calculated by:

$$\begin{bmatrix} x' \\ y' \\ w' \end{bmatrix} = H \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Then normalize:

$$(x', y') = \left(\frac{x'}{w'}, \frac{y'}{w'} \right)$$

ArUco Markers: ArUco markers are a type of fiducial marker—small, square patterns that can be easily detected in images and videos using computer vision algorithms, robust to occlusion. Each ArUco marker carries a unique ID encoded in its black and white pattern, allowing a vision system to recognize and distinguish between multiple markers in the same scene.

They are widely used in Camera calibration, Pose estimation (finding the position and orientation of the camera or object), Robot localization and navigation and Augmented reality (AR), Vision-guided robotic manipulation. They have built-in error detection and correction mechanism.

OpenCV supports ArUco as library **cv2.aruco**. We have implemented object detection and centroid calculation using this library.

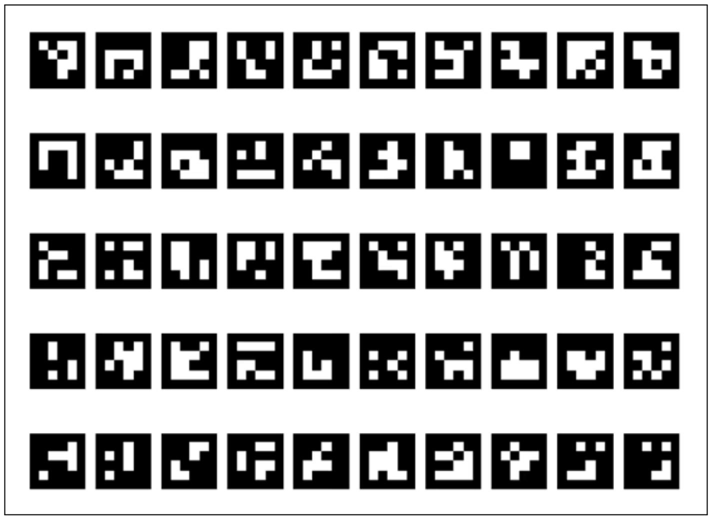

Figure 8: aruco.DICT 4X4 50

4.4 Coordinate Transformation and Real World Mapping

We define our robot's origin to be the point where the center of the base of the robot is kept, and from this point, the robot can extend it's links to reach any point that is within the ranges of the robot on the plane, it is kept. On the workspace, we draw a line joining the two bottom ArUco marker's center points and assume it as the x-axis of the workspace. We bisect this x-axis and draw a line perpendicular to x-axis from the bisection point and make it the y-axis of the workspace. We keep out robot at the origin of this coordinate system. Now our robot's origin coinsides with the workspace's origin and can operate on the workspace by operating in it's own coordinate system. By default the robot uses a Polar coordinate system, so necessary translations of the coordinates between Polar to Cartesian coordinate system can be done. Then for any given point on the workspace, it's Cartesian coordinate can be converted into the robot's parameters using inverse kinematics.

The registered image is resized in 900×700 resolution. The origin of the image is at the top-left corner of the image, and hence all the coordinates of the center points of the objects are with respect to the top-left corner. We have to convert it to the coordinate of the workspace. We identify the centers of the two ArUco markers at the bottom from the registered image and use the technique stated above to get the x-axis, y-axis and origin of the workspace in the registered image.
Let the robot's origin is (o_{x_R}, o_{y_R}) then we get the workspace coordinate of any pixel

point (x, y) of the image by given formula -

$$x' = x - o_{x_R}$$
$$y' = o_{y_R} - y$$

where, (x', y') is the workspace coordinate.

Here, we have to be careful that the obtained coordinates are also image coordinates.

Now to convert these image coordinates into real world units (e.g. cm) we use the following formula -

$$\text{cm\_per\_pixel\_x} = \frac{\text{paper\_width\_cm}}{\text{img\_width}}$$
$$\text{cm\_per\_pixel\_y} = \frac{\text{paper\_height\_cm}}{\text{img\_height}}$$
$$x_{cm} = x' \times \text{cm\_per\_pixel\_x}$$
$$y_{cm} = y' \times \text{cm\_per\_pixel\_y}$$

where,
x_{cm}, y_{cm} are x and y distances from respective axes in cm,
paper_width_cm = 29.7,
paper_height_cm = 21.0,
img_width = 900,
img_height = 700,
pixel coordinates (x, y).

4.5 Communication Setup

We have implemented this system in a WiFi network. In this setup we are using an android phone as a camera. We have connected the android phone, the computer and the ESP32 to the same WiFi network. The phone uses IP Webcam application to capture live camera feed and broadcast over a HTTP server. We are getting the input image from the phone over HTTP where the phone is the server and the main computer is the client. The computer is communicating with the ESP32 over a TCP socket. The ESP32 is the server and the computer is client. Over this connection the computer sends instructions to the robot which the robot executes.

5 Results

- The robot is successfully detecting objects within it's range in the workspace.
- It is picking up the objects from the detected locations in the workspace and keeping them in a fixed location.

For reference, a video is attached with this report.

6 Discussions

1. **Mechanical Error**:
    - There are errors in measuring the lengths of links of the robot.
    - The servo motors of the robot are not perfectly callibrated.
    - There is slight mis-alignment in robot due to handcrafting errors.
2. **Performance Error**:
    - The robot is performing reasonably good to reach towards the objects and pick up them perfectly in negative X-axis portion (Left side of the robot).
    - But, in positive X-axis portion (Right side of the robot) it is working little bit erroneous to reach towards the objects.
3. **Problem Faced and Solved**:
    - **Lagging of camera**: While live video feed from the camera, the camera was lagging due to incompatible network protocol (HTTP is not suitable for online video streaming. For which OpenCv is stagging old frame scenes in buffer).

      To solve this issue, we used capture once strategy in which we activate the camera, take an image, deactivate the camera. This ensures that the buffer is always clear and we always get the updated frame.
    - **Robot Tuning**: Since the robot is not properly alligned with the origin of the workspace due to handcrafting error, it is not very accurately reaching the objects locations in the workspace (Error of 0.6 - 0.7 cm maximum) sometimes.
      To solve this issue we have shifted the robot's origin slightly (2 - 3 mm)

towrds the right side of the robot. And also we have rotated the robot (2 - 3 degree) clockwise.

- **Improper Illumination**: As we are using contour detection method for object detection, the images are being converted into binary image using adaptive thresholding. This technique requires suffecient amount of illumination and minimized shadows of the objects.

  To solve this issue, We are applying proper illumination by using extra lights.

4. **Limitations**:

   - It is a serial manipulator so it has to work within it's range. It can't manipulate objects out of it's range.
   - It has some mechanical errors, so it works errorneous in some points.
   - It is an electronic device, so always a good, stable and suffecient amount of power supply is necessery.
   - Because of perspective warping the objects are also perspectively distorted in the registered image of the workspace and their detected centers are slightly(about 1.2mm) shifted from their real center, but these are neglected.
   - It works within it's specified and fixed environment(camera, workspace and network setup). Without this environment it can't work.
   - Because the system highly relies on ArUco markers to detect the workspace and define the coordinate systems, any error caused by them can result in a failure of the whole system.
   - The A4 paper of the workspace should not have any mark or stain as that can also be detected as an object by contour detection and cause serious error.

6.1 Conclusion and Future Works

This project successfully demonstrates the integration of computer vision with robotic manipulation to enable intelligent and autonomous interaction with real-world objects. By employing camera-based perception, ArUco marker-based workspace correction, object detection and robotic control, the system effectively detects, localizes and manipulates objects with a reasonable accuracy. The work lays a strong foundation for developing more advanced autonomous robotic systems and highlights the potential for further improvements in real-time performance, precision and versatility.

Future Works:

- Integration of Advanced Object Detection Models (ML and DL)
- Improvement in Grasp Planning.
- Dynamic Object Tracking.
- In place sorting of the objects of diferent sizes.
- Multi-Object Manipulation.
- Robustness to Lighting and Occlusion
- 3D Workspace Mapping and Environment Perception

References

[1] Md Tanzil Shahria, Md Samiul Haque Sunny, Md Ishrak Islam Zarif, Jawhar Ghommam, Sheikh Iqbal Ahamed, and Mohammad H Rahman, *A Comprehensive Review of Vision-Based Robotic Applications: Current State, Components, Approaches, Barriers, and Potential Solutions*, 2022.

[2] Ester Martinez-Martin and Angel P. Del Pobil, *Robot Vision for Manipulation: A Trip to Real-World Applications*, 2020.

[3] Md Tanzil Shahria, Jawhar Ghommam, Raouf Fareh, and Mohammad Habibur Rahman, *Vision-Based Object Manipulation for Activities of Daily Living Assistance Using Assistive Robot*, 2024.

[4] Umesh Kumar Sahu, Mebin K. S., Abhinav K., Muhammed Muzammil P, Ankur Jaiswal, Umesh Kumar Yadav, and Santanu Kumar Dash, *Autonomous Object Tracking with Vision-Based Control Using a 2-DOF Robotic Arm*, 2025.

[5] OpenCV Library Documentation – `https://docs.opencv.org`

[6] Garrido-Jurado et al., 2014, ArUco Marker Detection.

[7] ESP32 Microcontroller Datasheet – Espressif Systems.

[8] Robotic Arm Control using Servo Motors – `https://www.servodatabase.com`