---------- Software Design Description ----------

6.1 Introduction

This document presents the architecture and software design for the Vision-Guided Robotic Arm project. The system integrates computer visions and inverse-kinematics to allow our 3D printed robotic arm to detect, track, and interact with physical objects. The perception subsystem uses real time video footage from a USB webcam connected to Raspberry Pi 4, processed through NumPy and OpenCV for object detection. The control subsystem translates coordinates into motion commands while the actuation subsystem uses six servos to execute movements.

6.1.1 System Objectives

The objective of this system is to provide a vision-guided robotic arm that can detect, identify, and move physical objects with minimal user input. The system integrates computer vision and inverse kinematics to be able to recognize objects in the camera's field of view, and calculate the needed servo movements required to accurately grab and place objects. Objects of various shapes and colors are identified through an open computer vision library (OpenCV), and their positions are translated into coordinates for the robotic arm. Once the arm receives the coordinates, the arm will use inverse kinematics to accurately move its arm towards the desired object.

6.1.2 Hardware Interface

6.1.2.1 USB Webcam Interface

The Perception Subsystem uses a standard 2.0 camera connected directly to Raspberry Pi 4. It captures real-time 720p video frames at 30 fps.

6.1.2.2 Raspberry Pi 4 Interface

The Raspberry Pi 4 functions as the primary controller. It connects to the USB webcam for video capture and communicates with the motor-control board, which drives the servo motors.

6.1.2.3 Servo and Motor Control Board Interface

The motor-control board receives motion commands from the Raspberry Pi and transmits them to the STS3215 servos. Each servo is wired to a specific joint of the robotic arm allowing for physical movement.

6.1.2.4 Power Supply Interface

The system is powered by a DC power adapter that outputs 5V DC, which is suitable for the Raspberry PI and motor control components. The adapter connects through a barrel connector to the control board.

6.1.2.5 Development Hardware Interfaces

Development and debugging occur on personal laptops connected to the Raspberry Pi. These devices are used to upload Python scripts, run tests, and manage source code through GitHub.

6.1.2.6 3D-Printed Arm Assembly Interface

The 3D printed arm structure the physical framework for the robotic system. Its design was sourced from an open-source GitHub repository (SO-ARM100) to simplify assembly. The printed components house the servo motors and provide joints and linkages.

6.1.3 Software Interface

6.1.3.1 Programming Language Interface

All software is written in Python, running on Raspberry Pi.

6.1.3.2 Computer Vision Interface

The OpenCV library is used for object detection and image processing. Input frames from the webcam are passed through OpenCV functions.

6.1.3.3 Mathematical Computation Interface

NumPy is used to support matrix operations and inverse-kinematics calculations that determine servo positions.

6.1.3.4 Development Environment Interface

Development occurs within Visual Studio Code on personal laptops. The system's source code is stored in a GitHub Repo.

6.1.4 Human Interface

6.1.4.1 Developer Interface

Team members interact with the system through terminal commands on development computers or directly via Raspberry Pi 4.

6.1.4.2 Demonstration Interface

For demonstration purposes, users can observe the robotic arm's operation as it performs pick-and-place actions.

6.2 Architectural Design

The system architecture for the *Vision-Guided Robotic Arm* consists of three primary subsystems. These subsystems include:

1) Perception Subsystem
    - Captures real time video footage using a USB webcam connected to a Raspberry Pi 4.
    - Uses OpenCV and NumPy for object detection, color separation, and positional object analysis.
    - Outputs object coordinates in a normalized format to the control subsystem.
2) Control Subsystem
    - Implements inverse kinematics algorithms to calculate servo angles based on object coordinates.
    - In Python, this module interfaces with the motor control board to send position and motion commands.
    - Handles movement error correction, to ensure accurate motion trajectories.
3) Actuation Subsystem (Servo & Motor Hardware)
    - Includes six STS3215 servos, and a motor controller board.
    - Receives instructions and executes corresponding mechanical motion.
    - Provides feedback to control subsystem for calibration and adjustments.

6.2.1 Major Software Components

1) Vision Component:
    - Captures video frames, performs color filtering and object detection using OpenCV. Outputs the object coordinates.
2) Kinematics Component:

- Takes object coordinates and converts it into servo angles using inverse kinematics calculations.

3) Control Component:
   - Translates calculated angles into PWM commands for the motor controller.
4) Calibration Component:
   - Aligns servo's to their zero positions
5) Main Coordinator Component:
   - Handles startup and shutdown sequences

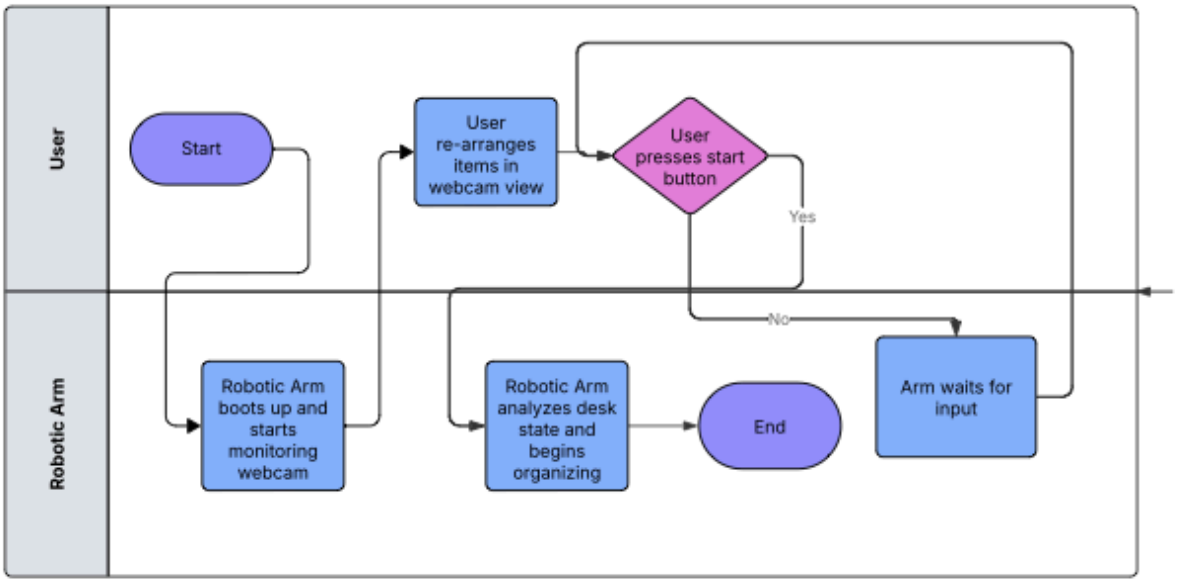
6.2.2 Major Software Interactions

The vision-guided robotic arm will have 5 major software interactions:

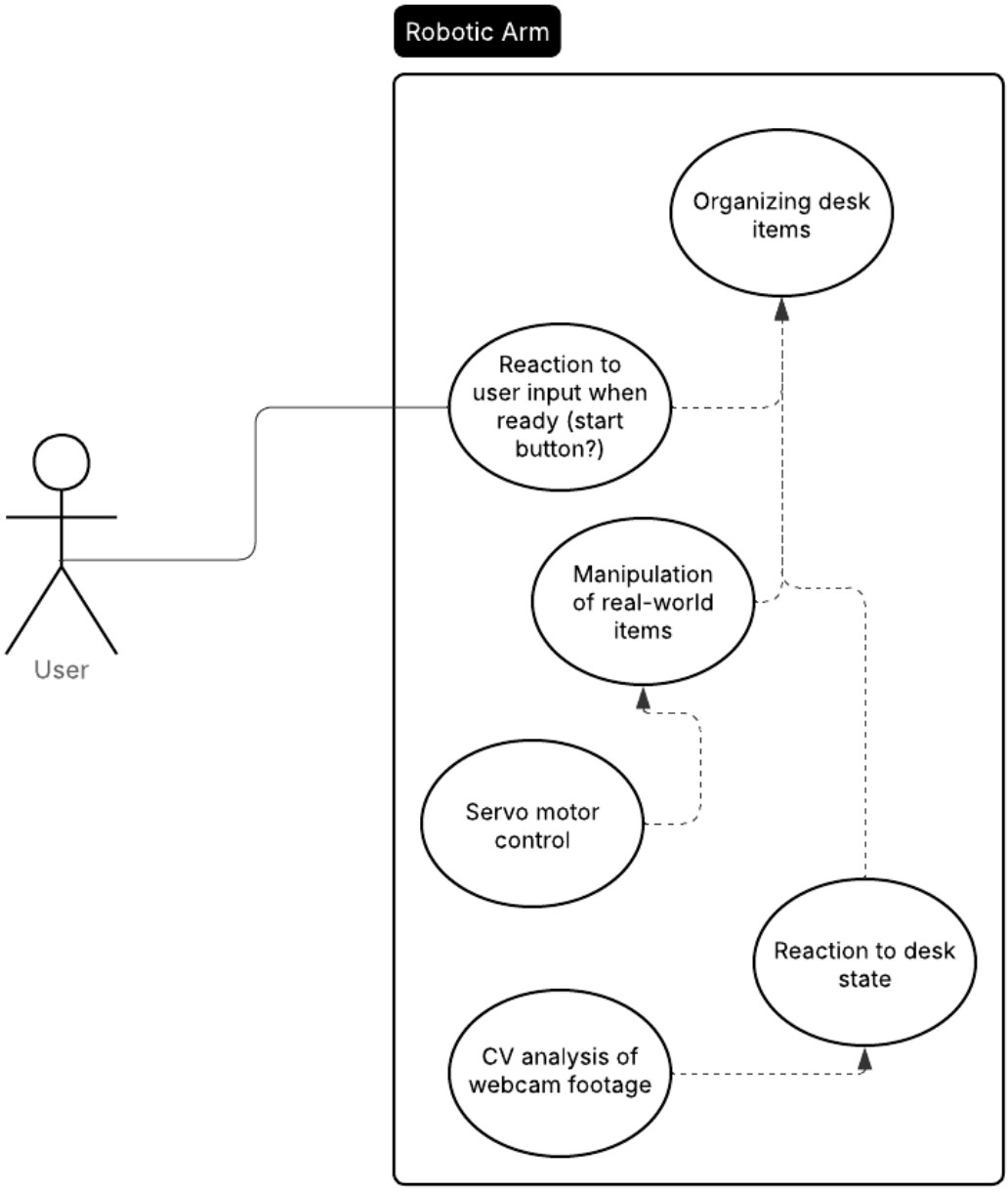
1. Vision to Kinematics:
   - The vision part outputs object coordinates as array: [x, y]. The kinematics part will take in these coordinates, and return a list of target servo angles.
2. Kinematics to Control:
   - The control portion receives the computed angles as a list [θ_0, θ_1, θ_2, θ_3, θ_4, θ_5]. These are translated into PWM values and issues I^2C commands to the motor board.
3. Control and Hardware:
   - Communication between the Raspberry Pi and the PWM board uses I^2C protocol. Ideal latency will be < 3 ms per command.
4. User Interaction:
   - The OpenCV display window runs on a separate thread to maintain responsive input handling and video feedback.

6.2.3 Architectural Design Diagrams

Swimlane Diagram



Use Case Diagram



Software Component Diagram