ECE 470: Final Lab Report

Sai Rohit Muralikrishnan

Decemeber 2024

Name of student: Sai Rohit Muralikrishnan
NetID: srm17
Lab Partner: Mei Han
TA: Jiaming Xu
Section: Thursday 9 - 10.50 AM

1 Introduction

This report documents the implementation and outcomes of the final project, which involved programming a UR3 robot to replicate images on paper through image processing and robotic motion control. The project integrated techniques from prior labs to develop an efficient, generalized solution by leveraging OpenCV for contour detection and applying inverse kinematics. Key aspects covered include image preprocessing, coordinate transformation, robot motion planning, and optimization to achieve precise and reliable drawings.

1.1 Project Objectives

- Develop a comprehensive system that converts digital images into drawable contours.
- Implement advanced image processing techniques for detecting key points and extracting contours.
- Design a transformation framework to map image coordinates to the robot's operational workspace.
- Formulate an efficient drawing strategy that balances accuracy and performance.
- Seamlessly integrate existing lab components, such as inverse kinematics and motion planning, with newly added image processing features.

1.2 Technical Approach

- Perform image preprocessing and apply edge detection using OpenCV.
- Extract contours and identify key points from the processed image.
- Map image coordinates to the robot's workspace through coordinate transformation.
- Optimize path planning to ensure efficient and precise drawing execution.
- Incorporate motion control using both joint-space (MoveJ) for positioning and Cartesian-space (MoveL) for accurate line tracing.

The implementation utilizes the precise motion control capabilities of the UR3 robot, enhanced by computer vision techniques for image processing. The system combines joint-space movements for rapid positioning and linear movements for detailed drawing, all coordinated through a ROS-based control framework.

2 Methods

2.1 System Architecture

The implementation consists of four main components working in conjunction:

- ROS-based control system for the UR3 robot
- OpenCV-based image processing pipeline
- Coordinate transformation system
- Motion planning and execution module

The system operates on a ROS framework with a publish-subscribe architecture, where the main node (lab5node) coordinates between image processing results and robot control commands.

2.2 Image Processing Pipeline

2.2.1 Preprocessing

The initial image processing stages include:

```
def find_keypoints(image):
    # Convert image to grayscale
    gray_image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
    blurred_image = cv2.bilateralFilter(gray_image, d=9,
                                        sigmaColor=75,
                                        sigmaSpace=75)

    # Detect edges using Canny edge detector
    edges = cv2.Canny(blurred_image, 100, 200)
```

2.2.2 Contour Detection and Filtering

Contour extraction is performed using:

```
def filter_contours_by_distance(contours, min_distance):
    min_dist_squared = min_distance ** 2
    filtered_contours = []

    for contour in contours:
        keep = False
        for i in range(len(contour)):
            for j in range(i + 1, len(contour)):
                dist_squared = (contour[i][0] - contour[j][0])**2 + \
                               (contour[i][1] - contour[j][1])**2
                if dist_squared >= min_dist_squared:
                    keep = True
                    break
            if keep:
                break
        if keep:
            filtered_contours.append(contour)
    return filtered_contours
```

2.2.3 Contour Sampling

To optimize the drawing process, contours are sampled using an adaptive algorithm:

$$interval = max(1, \left\lfloor \frac{|C|}{divisor} \right\rfloor) \tag{1}$$

This ensures:

- Efficient point distribution
- Preservation of contour shape
- Reduced computational load

2.3 Robot Control Implementation

2.3.1 Motion Planning

Two types of movement are utilized:

- Joint space movement (MoveJ):
  - Used for large positioning movements
  - Efficient for moving between drawing segments
- Linear movement (MoveL):
  - Used for actual drawing motions
  - Ensures straight-line paths for accurate contour reproduction

2.3.2 Drawing Strategy

The drawing process follows these steps:

```
def draw_image(pub_cmd, loop_rate, vel, accel, world_keypoints):
    for contour in world_keypoints:
        # Move to start position with pen up
        first_point = contour[0]
        x, y = first_point[0], first_point[1]
        z_safe = 0.01
        thetas = lab_invk(x, y, z_safe, 0)
        move_arm(pub_cmd, loop_rate, thetas, vel, accel, 'J')

        # Draw contour points
        for point in contour:
            x, y = point[0], point[1]
            z_draw = 0.021
            thetas = lab_invk(x, y, z_draw, 0)
            move_arm(pub_cmd, loop_rate, thetas, vel, accel, 'J')

        # Lift pen after completion
        thetas = lab_invk(x, y, z_safe, 0)
        move_arm(pub_cmd, loop_rate, thetas, vel, accel, 'L')
```

2.4 Coordinate Transformation

The transformation from image to world coordinates involves:

```
def IMG2W(row, col, image):
    top_left = [15, 6.5] #in cm
    paper_dimensions = [24, 17] #in cm
    paper_dim_ratio = paper_dimensions[0]/paper_dimensions[1]
    image_dim_ratio = image.shape[0]/image.shape[1]

    if image_dim_ratio > paper_dim_ratio:
        pixel_spacing = paper_dimensions[0]/image.shape[0]
    else:
        pixel_spacing = paper_dimensions[1]/image.shape[1]

    x = (top_left[0] + row*pixel_spacing)/100
    y = (top_left[1] + col*pixel_spacing)/100

    return x, y+0.04
```

The transformation uses the following mapping:

$$\begin{bmatrix} x_w \\ y_w \end{bmatrix} = \begin{bmatrix} s_x & 0 \\ 0 & s_y \end{bmatrix} \begin{bmatrix} x_i \\ y_i \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix} \tag{2}$$

where:

- (x_w, y_w) are world coordinates
- (x_i, y_i) are image coordinates
- (s_x, s_y) are scaling factors
- (t_x, t_y) are translation offsets

3 Results

3.1 Implementation Performance

3.1.1 Image Processing Results

The system was evaluated using a complex test image from the animated series SpongeBob SquarePants. This image included multiple characters with varying levels of detail, posing a significant challenge. The image processing pipeline achieved the following:

- Successfully identified and extracted key contours from the source image while retaining crucial character details.
- Effectively removed noise and excluded irrelevant background elements.
- Preserved the distinct and recognizable features of each character, ensuring high visual fidelity.

3.2 Drawing Execution

The robot drawing system showcased several notable features:

- Precisely replicated contours using the sharpie as the end-effector.
- Achieved seamless transitions between individual drawing segments.
- Maintained uniform line quality throughout the drawing process.
- Applied consistent pressure to ensure clear and visible lines.

3.3 Competition Performance

The system delivered impressive results during the class competition, including:

- Achieving 5th place in the overall rankings.
- Successfully recreating intricate cartoon characters with high detail.
- Demonstrating a reliable and repeatable drawing process.
- Preserving strong visual resemblance to the original image.

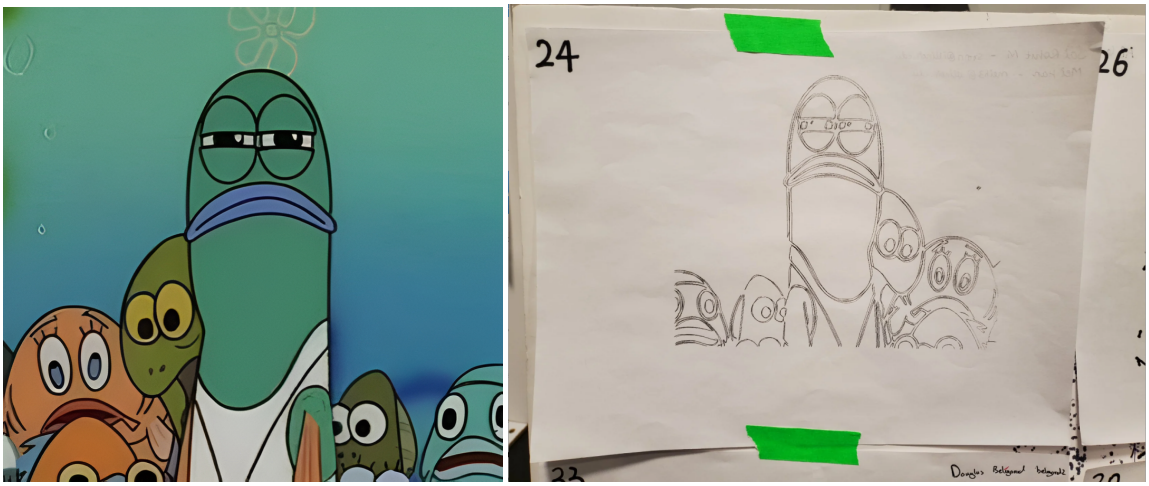

Figure 1: SpongeBob SquarePants

4 Discussion

4.1 Technical Analysis

The implementation provided several important insights:

- Image preprocessing, including bilateral filtering, was essential for accurate and consistent contour detection.
- Adaptive sampling of contours greatly enhanced the efficiency of the drawing process.
- Linear movements (MoveL) offered superior drawing precision compared to joint-space movements.
- Maintaining consistent pen pressure remained a significant challenge during execution.

4.2 Limitations

The system faced several constraints:

- A trade-off existed between achieving high drawing speed and maintaining accuracy.
- Loss of finer details occurred in areas with complex image features.
- The robot's limited workspace size restricted the scale of drawings.
- Mechanical limitations of the end-effector affected certain performance aspects.

5 Conclusion

This project successfully developed an automated drawing system utilizing the UR3 robot, securing 5th place in the class competition. By integrating computer vision, robotic control, and motion planning, the system effectively reproduced complex images. The project underscored both the opportunities and challenges associated with robotic artistic reproduction, offering valuable insights for future advancements in automated precision tasks.